CONJUNTO de DELITOS

V0

Generado por Doxygen 1.8.6

Domingo, 15 de Noviembre de 2015 18:26:33

Índice

1. Documentación Práctica

Versión

v0

Autor

Juan F. Huete

1.1. Introducción

En esta practica se pretende avanzar en el uso de las estructuras de datos, para ello comenzaremos con el diseño de distintos tipos de datos que nos permitan manejar la información asociada a la base de datos de delitos de la ciudad de Chicago (EEUU)

1.1.1. Conjunto de Datos

El conjunto de datos con el que trabajaremos es un subconjunto de la base de datos de la City of Chicago, "-Crimes-2001 to present" los informes sobre delitos (con la excepción de asesinatos) que han ocurrido en la ciudad de Chicago (EEUU) desde 2001 hasta el presente (menos la última semana). Los datos son extraidos del "Chicago Police Department's CLEAR (Citizen Law Enforcement Analysis and Reporting)". La base de datos original, con unos 6 millones de delitos, se puede obtener entre otros en formato csv (del inglés comma-separated values, que representa una tabla, en las que las columnas se separan por comas y las filas por saltos de línea. Así, la primera línea del fichero indica los campos de la base de datos, y el resto de líneas la descripción asociada a cada delito,

```
ID,Case Number,Date,Block,IUCR,Primary Type,Description,Location Description,Arr
est,Domestic,Beat,District,Ward,Community Area,FBI Code,X Coordinate,Y Coordinat
e,Year,Updated On,Latitude,Longitude,Location
10230953,HY418703,09/10/2015 11:56:00 PM,048XX W NORTH AVE,0498,BATTERY,AGGRAVAT
ED DOMESTIC BATTERY: HANDS/FIST/FEET SERIOUS INJURY,APARTMENT,true,true,2533,025
,37,25,04B,1143637,1910194,2015,09/17/2015 11:37:18 AM,41.909605035,-87.74777714
5,"(41.909605035, -87.747777145)"
10230979,HY418750,09/10/2015 11:55:00 PM,120XX S PARNELL AVE,0486,BATTERY,DOMEST
IC BATTERY SIMPLE,ALLEY,true,true,0523,005,34,53,08B,1174806,1825089,2015,09/17/
2015 11:37:18 AM,41.675427135,-87.63581257,"(41.675427135, -87.63581257)"
10231208,HY418843,09/10/2015 11:50:00 PM,021XX W BERWYN AVE,0820,THEFT,$500 AND
UNDER,STREET,false,false,2012,020,40,4,06,1161036,1935171,2015,09/17/2015 11:37:
18 AM,41.97779966,-87.683164484,"(41.97779966, -87.683164484)"
```

1.2. "Fecha Límite de Entrega"

C++ no tiene un tipo propio para trabajar con fechas, por lo que debemos implementar la clase fecha que deberá tener entre otros los métodos abajo indicados. La especificación de la clase fecha se realizará en el fichero fecha.h y la implementación de la clase fecha la haremos en el fichero fecha.hxx.

```
class fecha {
private:
  int  sec;   // seconds of minutes from 0 to 61
  int  min;   // minutes of hour from 0 to 59
  int  hour;  // hours of day from 0 to 24
  int  mday;  // day of month from 1 to 31
  int  mon;   // month of year from 0 to 11
  int  year;  // year since 2000

public:
 fecha (); //Constructor de fecha por defecto
 fecha (const string & s); // s es un string con el formato mm/dd/aaaa  hh:mm:ss AM/PM

 fecha & operator=(const fecha & f);
 fecha & operator=(const string & s); // s es un string con el formato mm/dd/aaaa hh:mm:ss
       AM/PM
 string toString() const;

// Operadores relacionales
  bool operator==(const fecha & f) const;
 bool operator<(const fecha & f)const;
 bool operator>(const fecha & f) const;
 bool operator<=(const fecha & f)const;
 bool operator>=(const fecha & f) const;
 bool operator!=(const fecha & f)const;
}

 ostream& operator« ( ostream& os, const fecha & f); imprime
      fecha con el formato  mm/dd/aaaa hh:mm:ss AM/PM

#include "fecha.hxx" // Incluimos la implementacion.
```

Así, podremos trabajar con fechas como indica el siguiente código

```
...
fecha f1;
f1 = "09/10/2015 11:55:00 PM";
fecha f2(f1);
...
fecha f3 = "09/04/2010 11:55:00 PM"
..
if (f1<f3)
  cout « f1 « " es menor que " f3;
...
```

1.3. Crimen

A igual que con la clase fecha, la especificación del tipo crimen y su implementación se realizará en los ficheros crimen.h y crimen.hxx, respectivamente, y debe tener la información de los atributos (con su representacion asociada)

- ID: identificador del delito (long int)
- Case Number: Código del caso (string)
- Date: Fecha en formato mm/dd/aaaa hh:mm:ss AM/PM (fecha, ver arriba)
- IUCR: Código del tipo de delito según Illinois Uniform Crime Reporting, IUCR (string)
- Primary Type: Tipo de delito (string)
- Description: Descripción más detallada (string)
- Location Description: Descripción del tipo de localización (string)
- Arrest: Si hay arrestos o no (boolean)

- Domestic: Si es un crimen domenstico o no (boolean)
- Latitude: Coordenada de latitud (double)
- Longitude: Coordenad de longitud (double)

```
// Fichero crimen.h
class crimen {
 ....
}

#include "crimen.hxx" // Incluimos la implementacion
```

1.4. Conjunto como TDA contenedor de información

Nuestro conjunto será un contenedor que permite almacenar la información de la base de datos de delitos. Para un mejor acceso, los elementos deben estar ordenados según ID, en orden creciente. Como TDA, lo vamos a dotar de un conjunto restringido de métodos (inserción de elementos, consulta de un elemento por ID, etc.). Este diccionario "simulará" un set de la stl, con algunas claras diferencias pues, entre otros, no estará dotado de la capacidad de iterar (recorrer) a través de sus elementos, que se hará en las siguientes prácticas.

Asociado al conjunto, tendremos los tipos

```
conjunto::entrada
```

```
conjunto::size_type
```

que permiten hacer referencia a los elementos almacenados en cada una de las posiciones y el número de elementos del mismo, respectivamente.

1.5. "Se Entrega / Se Pide"

1.5.1. Se entrega

En esta práctica se entrega los fuentes necesarios para generar la documentación de este proyecto así como el código necesario para resolver este problema. En concreto los ficheros que se entregan son:

- documentacion.pdf Documentación de la práctica en pdf.
- dox_diccionario Este fichero contiene el fichero de configuración de doxigen necesario para generar la documentación del proyecto (html y pdf). Para ello, basta con ejecutar desde la línea de comando

  ```
  doxygen doxPractica
  ```

  La documentación en html la podemos encontrar en el fichero ./html/index.html, para generar la documentación en latex es suficiente con hacer los siguientes pasos

  ```
  cd latex
  make
  ```

  como resultado tendremos el fichero refman.pdf que incluye toda la documentación generada.

- conjunto.h Especificación del TDA conjunto.
- conjunto.hxx plantilla de fichero donde debemos implementar el conjunto.
- crimen.h Plantilla para la especificación del TDA crimen
- crimen.hxx plantilla de fichero donde debemos implementar el crimen
- fecha.h Plantilla para la especificación del TDA fecha
- fecha.hxx plantilla de fichero donde debemos implementarlo
- principal.cpp fichero donde se incluye el main del programa. En este caso, se toma como entrada el fichero de datos "crimenes.csv" y se debe cargar en el set.

1.5.2. Se Pide

- Diseñar la función de abstracción e invariante de la representación del tipo fecha
- Diseñar la función de abstracción e invariante de la representación del tipo crimen.
- Se pide implementar el código asociado a los ficheros .hxx.
- Analizar la eficiencia teórica y empírica de las operaciones de insercion y búsqueda en el conjunto.

1.6. Representación

El alumno deberá realizar una implementación utilizando como base el TDA vector de la STL. En particular, la representación que se utiliza es un vector ordenado de entradas, teniendo en cuenta el valor de la clave ID.

1.6.1. Función de Abstracción :

Función de Abstracción: AF: Rep => Abs

```
dado C =(vector<crimen> vc ) ==> Conjunto BD;
```

Un objeto abstracto, BD, representando una colección ORDENADA de crimenes según ID, se instancia en la clase conjunto como un vector ordenado de crimenes,

1.6.2. Invariante de la Representación:

Propiedades que debe cumplir cualquier objeto

```
BD.size() == C.vc.size();

Para todo i, 0 <= i < V.vc.size() se cumple
    C.vc[i].ID > 0;
Para todo i, 0 <= i < D.dic.size()-1 se cumple
    C.vc[i].ID<= D.dic[i+1].ID
```

1.7. "Fecha Límite de Entrega"

La fecha límite de entrega será el 6 de Noviembre.

2. Lista de tareas pendientes

Clase fecha

Escribe la documentación de la clase

Implementar esta clase

Miembro operator<< (ostream &sal, const conjunto &D)

implementar esta funcion

3. Índice de clases

3.1. Lista de clases

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

4. Indice de archivos

4.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

5. Documentación de las clases

5.1. Referencia de la Clase conjunto::arrest_iterator

class arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!=

`#include <conjunto.h>`

Métodos públicos

- arrest_iterator ()
  Constructor por defecto del iterador.
- arrest_iterator (const arrest_iterator &it)
  Constructor de copia.
- const conjunto::entrada & operator∗ () const
  Operador ∗ para acceder al contenido del iterador.
- arrest_iterator operator++ (int)
  Operador ++ para el postincremento del iterador.
- arrest_iterator & operator++ ()
  Operador ++ para el preincremento.
- arrest_iterator & operator-- ()
  Operador – para el predecremento del iterador.
- arrest_iterator operator-- (int)
  Operador – para el postdecremento del iterador.
- bool operator== (const arrest_iterator &it)
  Operador == que comprueba si dos iteradores son iguales.
- bool operator!= (const arrest_iterator &it)
  Operador != que comprueba si dos iteradores son distintos.
- arrest_iterator & operator= (const arrest_iterator &it)
  Operador de asignacion.

Atributos privados

- const vector< entrada > ∗ mi_conj
- vector< conjunto::entrada >
  ::iterator itv

Amigas

- class conjunto

5.1.1. Descripción detallada

class arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!=

5.1.2. Documentación del constructor y destructor

5.1.2.1. conjunto::arrest_iterator::arrest_iterator ( )

Constructor por defecto del iterador.

5.1.2.2. conjunto::arrest_iterator::arrest_iterator (const arrest_iterator & *it*)

Constructor de copia.

Parámetros

in	*it*	

5.1.3. Documentación de las funciones miembro

5.1.3.1. bool conjunto::arrest_iterator::operator!= (const **arrest_iterator** & *it*)

Operador != que comprueba si dos iteradores son distintos.

Parámetros

in	*it*	iterador a comparar

5.1.3.2. const **conjunto::entrada**& conjunto::arrest_iterator::operator∗ ( ) const

Operador ∗ para acceder al contenido del iterador.

5.1.3.3. **arrest_iterator** conjunto::arrest_iterator::operator++ (int)

Operador ++ para el postincremento del iterador.

5.1.3.4. **arrest_iterator**& conjunto::arrest_iterator::operator++ ( )

Operador ++ para el preincremento.

5.1.3.5. **arrest_iterator**& conjunto::arrest_iterator::operator-- ( )

Operador – para el predecremento del iterador.

5.1.3.6. **arrest_iterator** conjunto::arrest_iterator::operator-- (int)

Operador – para el postdecremento del iterador.

5.1.3.7. **arrest_iterator**& conjunto::arrest_iterator::operator= (const **arrest_iterator** & *it*)

Operador de asignacion.

Parámetros

in	*it*	iterador a asignar

5.1.3.8. bool conjunto::arrest_iterator::operator== (const **arrest_iterator** & *it*)

Operador == que comprueba si dos iteradores son iguales.

Parámetros

in	*it*	iterador a comparar

5.1.4. Documentación de las funciones relacionadas y clases amigas

5.1.4.1. friend class **conjunto** [friend]

5.1.5. Documentación de los datos miembro

5.1.5.1. vector<**conjunto::entrada**>::**iterator** conjunto::arrest_iterator::itv [private]

5.1.5.2. const vector<entrada>* conjunto::arrest_iterator::mi_conj [private]

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

- conjunto.h

5.2. Referencia de la Clase conjunto

Clase conjunto.

`#include <conjunto.h>`

Clases

- class arrest_iterator

  class arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator, operator++, operator++(int) operator=, operator==, operator!=*
- class const_arrest_iterator

  class const_arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator, operator++, operator++(int) operator=, operator==, operator!=*
- class const_description_iterator

  class const_description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion*
- class const_iterator

  class const_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator, operator++, operator++(int) operator=, operator==, operator!=*
- class description_iterator

  class description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion*
- class iterator

  class iterator forward iterador sobre el conjunto, LECTURA iterator() ,operator(), operator++, operator++(int) operator=, operator==, operator!=*

Tipos públicos

- typedef crimen entrada

  entrada permite hacer referencia al elemento almacenados en cada una de las posiciones del conjunto
- typedef unsigned int size_type

  size_type numero de elementos en el conjunto

Métodos públicos

- conjunto ()

  constructor primitivo.
- conjunto (const conjunto &d)

  constructor de copia
- iterator find (const long int &id)

  busca un crimen en el conjunto
- const_iterator find (const long int &id) const

  busca un crimen en el conjunto
- conjunto findIUCR (const string &iucr) const

busca los crimenes con el mismo codigo IUCR

- conjunto findDESCR (const string &descr) const

  busca los crimenes que contienen una determinada descripcion

- bool insert (const conjunto::entrada &e)

  Inserta una entrada en el conjunto.

- bool erase (const long int &id)

  Borra el delito dado un identificacador.

- bool erase (const conjunto::entrada &e)

  Borra una crimen con identificador dado por e.getID() en el conjunto. Busca la entrada con id en el conjunto (o e.getID() en el segundo caso) y si la encuentra la borra.

- conjunto & operator= (const conjunto &org)

  operador de asignacion

- size_type size () const

  numero de entradas en el conjunto

- bool empty () const

  Chequea si el conjunto esta vacio.

- iterator begin ()

  devuelve iterador al inicio del conjunto

- iterator end ()

  devuelve iterador al final (posicion siguiente al ultimo del conjunto)

- const_iterator cbegin () const
- const_iterator cend () const

  iterador al final

- description_iterator dbegin (const string &descr)

  devolver primera posicion del elemento que empareja con la descripcion descr

- description_iterator dend ()

  devolver fin del conjunto

- const_description_iterator cdbegin (const string &descr) const

  devolver primera posicion del elemento que empareja con la descripcion descr

- const_description_iterator cdend () const

  devolver fin del conjunto

- arrest_iterator abegin ()

  devuelve iterador al primer crimen con arresto del conjunto

- arrest_iterator aend ()

  devuelve iterador al final (posicion siguiente al ultimo del conjunto)

- const_arrest_iterator cabegin () const

  devuelve iterador constante al primer crimen con arresto del conjunto

- const_arrest_iterator caend () const

  devuelve iterador constante al final (posicion siguiente al ultimo del conjunto)

Métodos privados

- bool cheq_rep () const

  Chequea el Invariante de la representacion.

Atributos privados

- vector< crimen > vc

Amigas

- class iterator
- class const_iterator
- ostream & operator<< (ostream &sal, const conjunto &D)

  imprime todas las entradas del conjunto

5.2.1. Descripción detallada

Clase conjunto.

Metodos–> conjunto:: conjunto(), insert(), find(), findIUCR(), findDESCR(), erase(), size(), empty()

Tipos–> conjunto::entrada, conjunto::size_type

Descripcion

Un conjunto es un contenedor que permite almacenar en orden creciente un conjunto de elementos no repetidos. En nuestro caso el conjunto va a tener un subconjunto restringido de metodos (insercion de elementos, consulta de un elemento, etc). Este conjunto "simulara" un conjunto de la stl, con algunas claras diferencias pues, entre otros, no estara dotado de la capacidad de iterar (recorrer) a traves de sus elementos.

Asociado al conjunto, tendremos el tipo

```
conjunto::entrada
```

que permite hacer referencia al elemento almacenados en cada una de las posiciones del conjunto, en nuestro caso delitos (crimenes). Para esta entrada el requisito es que tenga definidos el operador< y operator=

Ademas encontraremos el tipo

```
conjunto::size_type
```

que permite hacer referencia al numero de elementos en el conjunto.

El numero de elementos en el conjunto puede variar dinamicamente; la gestion de la memoria es automatica.

Ejemplo de su uso:

```
...
conjunto DatosChicago, agresion;
crimen cr;

conjunto.insert(cr);
...
agresion = conjunto.findDESCR("BATTERY");

if (!agresion.empty()){
 cout «"Tenemos "« agresion.size() « " agresiones" « endl;
 cout « agresion « endl;
} else "No hay agresiones en el conjunto" « endl;
...
```

5.2.2. Documentación de los 'Typedef' miembros de la clase

5.2.2.1. typedef crimen conjunto::entrada

entrada permite hacer referencia al elemento almacenados en cada una de las posiciones del conjunto

5.2.2.2. typedef unsigned int conjunto::size_type

size_type numero de elementos en el conjunto

5.2.3. Documentación del constructor y destructor

5.2.3.1. conjunto::conjunto ( )

constructor primitivo.

5.2.3.2. conjunto::conjunto (const conjunto & *d*)

constructor de copia

Parámetros

in	*d*	conjunto a copiar

5.2.4. Documentación de las funciones miembro

5.2.4.1. arrest_iterator conjunto::abegin ( )

devuelve iterador al primer crimen con arresto del conjunto

5.2.4.2. arrest_iterator conjunto::aend ( )

devuelve iterador al final (posicion siguiente al ultimo del conjunto)

5.2.4.3. iterator conjunto::begin ( )

devuelve iterador al inicio del conjunto

5.2.4.4. const_arrest_iterator conjunto::cabegin ( ) const

devuelve iterador constante al primer crimen con arresto del conjunto

5.2.4.5. const_arrest_iterator conjunto::caend ( ) const

devuelve iterador constante al final (posicion siguiente al ultimo del conjunto)

5.2.4.6. const_iterator conjunto::cbegin ( ) const

Devuelve

Devuelve el const_iterator a la primera posicion del conjunto.

Postcondición

no modifica el diccionario

5.2.4.7. const_description_iterator conjunto::cdbegin (const string & *descr*) const

devolver primera posicion del elemento que empareja con la descripcion descr

Parámetros

in	*descr*	descripcion de buscamos

Devuelve

un iterador constante que apunta a la primera posicion, el emparejamiento se hace teniendo en cuenta que descr debe ser una subcadena de la descripcion del delito.

5.2.4.8. const_description_iterator conjunto::cdend ( ) const

devolver fin del conjunto

Devuelve

un iterador constante que apunta a la posicion final

5.2.4.9. const_iterator conjunto::cend ( ) const

iterador al final

Devuelve

Devuelve el iterador constante a la posicion final del conjunto.

Postcondición

no modifica el diccionario

5.2.4.10. bool conjunto::cheq_rep ( ) const [private]

Chequea el Invariante de la representacion.

Invariante

IR: rep ==> bool

- Para todo i, 0 <= i < vc.size() se cumple vc[i].ID > 0;
- Para todo i, 0 <= i <= D.dic.size()-1 se cumple vc[i].ID< vc[i+1].ID

Devuelve

true si el invariante es correcto, falso en caso contrario

5.2.4.11. description_iterator conjunto::dbegin (const string & *descr*)

devolver primera posicion del elemento que empareja con la descripcion descr

Parámetros

in	*descr*	descripcion de buscamos

Devuelve

un iterador que apunta a la primera posicion, el emparejamiento se hace teniendo en cuenta que descr debe ser una subcadena de la descripcion del delito.

5.2.4.12. description_iterator conjunto::dend ( )

devolver fin del conjunto

Devuelve

un iterador que apunta a la posicion final

5.2.4.13. bool conjunto::empty ( ) const

Chequea si el conjunto esta vacio.

Devuelve

true si size()==0, false en caso contrario.

5.2.4.14. iterator conjunto::end ( )

devuelve iterador al final (posicion siguiente al ultimo del conjunto)

5.2.4.15. bool conjunto::erase (const long int & *id*)

Borra el delito dado un identificacador.

Busca la entrada con id en el conjunto y si la encuentra la borra

Parámetros

in	*id*	a borrar

Devuelve

true si la entrada se ha podido borrar con exito. False en caso contrario

Postcondición

Si esta en el conjunto su tamaño se decrementa en 1.

5.2.4.16. bool conjunto::erase (const conjunto::entrada & *e*)

Borra una crimen con identificador dado por e.getID() en el conjunto. Busca la entrada con id en el conjunto (o e.getID() en el segundo caso) y si la encuentra la borra.

Parámetros

in	*entrada*	con e.getID() que geremos borrar, el resto de los valores no son tenidos en cuenta

Devuelve

true si la entrada se ha podido borrar con exito. False en caso contrario

Postcondición

Si esta en el conjunto su tamaño se decrementa en 1.

5.2.4.17. iterator conjunto::find (const long int & *id*)

busca un crimen en el conjunto

Parámetros

id	identificador del crimen buscar

Devuelve

Si existe una entrada en el conjunto devuelve un iterador a la posicion donde esta el elemento. Si no se encuentra devuelve end()

Postcondición

no modifica el conjunto.

Ejemplo

```
if (C.find(12345)!=C.end() ) cout « "Esta" ;
else cout « "No esta";
```

5.2.4.18. const_iterator conjunto::find (const long int & *id*) const

busca un crimen en el conjunto

Parámetros

id	identificador del crimen buscar

Devuelve

Si existe una entrada en el conjunto devuelve un iterador a lo posicion donde esta el elemento. Si no se encuentra devuelve end()

Postcondición

no modifica el conjunto.

Ejemplo

```
if (C.find(12345)!=C.end() ) cout « "Esta" ;
else cout « "No esta";
```

5.2.4.19. conjunto conjunto::findDESCR (const string & *descr*) const

busca los crimenes que contienen una determinada descripcion

Parámetros

descr	string que representa la descripcion del delito buscar

Devuelve

Devuelve un conjunto con todos los crimenes que contengan descr en su descripcionR. Si no existe ninguno devuelve el conjunto vacio.

Postcondición

no modifica el conjunto.

```
Uso
    vector<crimen> C, A;
    ....
    A = C.findDESCR("BATTERY");
```

5.2.4.20. conjunto conjunto::findIUCR (const string & *iucr*) const

busca los crimenes con el mismo codigo IUCR

Parámetros

iucr	identificador del crimen buscar

Devuelve

Devuelve un conjunto con todos los crimenes con el codigo IUCR. Si no existe ninguno devuelve el conjunto vacio.

Postcondición

no modifica el conjunto.

```
Uso
    vector<crimen> C, A;
    ....
    A = C.findIUCR("0460");
```

5.2.4.21. bool conjunto::insert (const conjunto::entrada & *e*)

Inserta una entrada en el conjunto.

Parámetros

e	entrada a insertar

Devuelve

true si la entrada se ha podido insertar con exito. False en caso contrario

Postcondición

Si e no esta en el conjunto, el size() sera incrementado en 1.

5.2.4.22. conjunto& conjunto::operator= (const conjunto & *org*)

operador de asignacion

Parámetros

in	*org*	conjunto a copiar. Crea un conjunto duplicado exacto de org.

5.2.4.23. size_type conjunto::size ( ) const

numero de entradas en el conjunto

Postcondición

No se modifica el conjunto.

5.2.5. Documentación de las funciones relacionadas y clases amigas

5.2.5.1. friend class const_iterator [friend]

5.2.5.2. friend class iterator [friend]

5.2.5.3. ostream& operator<< (ostream & *sal,* const conjunto & *D*) [friend]

imprime todas las entradas del conjunto

Postcondición

No se modifica el conjunto.

Tareas pendientes implementar esta funcion

5.2.6. Documentación de los datos miembro

5.2.6.1. vector<crimen> conjunto::vc [private]

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

- conjunto.h

5.3. Referencia de la Clase conjunto::const_arrest_iterator

class const_arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!=

`#include <conjunto.h>`

Métodos públicos

- const_arrest_iterator ()
- const_arrest_iterator (const const_arrest_iterator &it)
- const_arrest_iterator (const arrest_iterator &it)

  Convierte arrest_iterator en const_arrest_iterator.
- const conjunto::entrada & operator∗ () const

  Operador ∗ para acceder al contenido del iterador.
- const_arrest_iterator operator++ (int)

  Operador ++ para el postincremento del iterador.
- const_arrest_iterator & operator++ ()

  Operador ++ para el preincremento.
- const_arrest_iterator & operator-- ()

  Operador – para el predecremento del iterador.
- const_arrest_iterator operator-- (int)

  Operador – para el postdecremento del iterador.
- bool operator== (const const_arrest_iterator &it)

  Operador == que comprueba si dos iteradores son iguales.
- bool operator!= (const const_arrest_iterator &it)

  Operador != que comprueba si dos iteradores son distintos.
- const_arrest_iterator & operator= (const const_arrest_iterator &it)

  Operador de asignacion.

Atributos privados

- const vector< entrada > ∗ mi_conj
- vector< conjunto::entrada > ::const_iterator c_itv

Amigas

- class conjunto

5.3.1. Descripción detallada

class const_arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!=

5.3.2. Documentación del constructor y destructor

5.3.2.1. conjunto::const_arrest_iterator::const_arrest_iterator ( )

5.3.2.2. conjunto::const_arrest_iterator::const_arrest_iterator (const const_arrest_iterator & *it*)

5.3.2.3. conjunto::const_arrest_iterator::const_arrest_iterator (const arrest_iterator & *it*)

Convierte arrest_iterator en const_arrest_iterator.

5.3.3. Documentación de las funciones miembro

5.3.3.1. bool conjunto::const_arrest_iterator::operator!= (const const_arrest_iterator & *it*)

Operador != que comprueba si dos iteradores son distintos.

Parámetros

in	*it*	iterador a comparar

5.3.3.2. const conjunto::entrada& conjunto::const_arrest_iterator::operator∗ ( ) const

Operador ∗ para acceder al contenido del iterador.

5.3.3.3. const_arrest_iterator conjunto::const_arrest_iterator::operator++ (int)

Operador ++ para el postincremento del iterador.

5.3.3.4. const_arrest_iterator& conjunto::const_arrest_iterator::operator++ ( )

Operador ++ para el preincremento.

5.3.3.5. const_arrest_iterator& conjunto::const_arrest_iterator::operator-- ( )

Operador – para el predecremento del iterador.

5.3.3.6. const_arrest_iterator conjunto::const_arrest_iterator::operator-- (int)

Operador – para el postdecremento del iterador.

5.3.3.7. const_arrest_iterator& conjunto::const_arrest_iterator::operator= (const const_arrest_iterator & *it*)

Operador de asignacion.

Parámetros

in	*it*	iterador a asignar

5.3.3.8. bool conjunto::const_arrest_iterator::operator== (const const_arrest_iterator & *it*)

Operador == que comprueba si dos iteradores son iguales.

Parámetros

in	*it*	iterador a comparar

5.3.4. Documentación de las funciones relacionadas y clases amigas

5.3.4.1. friend class conjunto [friend]

5.3.5. Documentación de los datos miembro

5.3.5.1. vector<conjunto::entrada>::const_iterator conjunto::const_arrest_iterator::c_itv [private]

5.3.5.2. const vector<entrada>* conjunto::const_arrest_iterator::mi_conj [private]

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

- conjunto.h

5.4. Referencia de la Clase conjunto::const_description_iterator

class const_description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator*, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion

`#include <conjunto.h>`

Métodos públicos

- const_description_iterator ()

  Constructor por defecto del iterador.
- const_description_iterator (const const_description_iterator &it)

  Constructor de copia.
- const_description_iterator (const description_iterator &it)

  Convierte description_iterator en const_description_iterator.
- const conjunto::entrada & operator* () const

  *Operador * para acceder al contenido del iterador.*
- const_description_iterator operator++ (int)

  Operador ++ para el postincremento del iterador.
- const_description_iterator & operator++ ()

  Operador ++ para el preincremento.
- const_description_iterator operator-- (int)

  Operador – para el postdecremento del iterador.
- const_description_iterator & operator-- ()

  Operador – para el predecremento del iterador.
- bool operator== (const const_description_iterator &it)

  Operador == que comprueba si dos iteradores son iguales.
- bool operator!= (const const_description_iterator &it)

  Operador != que comprueba si dos iteradores son distintos.
- const_description_iterator & operator= (const const_description_iterator &it)

  Operador de asignacion.

Atributos privados

- const vector< entrada > * mi_conj
- string descr
- vector< conjunto::entrada > ::const_iterator c_itv

Amigas

- class conjunto

5.4.1. Descripción detallada

class const_description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator*, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion

5.4.2. Documentación del constructor y destructor

5.4.2.1. conjunto::const_description_iterator::const_description_iterator ( )

Constructor por defecto del iterador.

5.4.2.2. conjunto::const_description_iterator::const_description_iterator (const const_description_iterator & *it*)

Constructor de copia.

Parámetros

in	*it*	

5.4.2.3. conjunto::const_description_iterator::const_description_iterator (const description_iterator & *it*)

Convierte description_iterator en const_description_iterator.

5.4.3. Documentación de las funciones miembro

5.4.3.1. bool conjunto::const_description_iterator::operator!= (const const_description_iterator & *it*)

Operador != que comprueba si dos iteradores son distintos.

Parámetros

in	*it*	iterador a comparar

5.4.3.2. const conjunto::entrada& conjunto::const_description_iterator::operator* ( ) const

Operador * para acceder al contenido del iterador.

5.4.3.3. const_description_iterator conjunto::const_description_iterator::operator++ (int)

Operador ++ para el postincremento del iterador.

5.4.3.4. const_description_iterator& conjunto::const_description_iterator::operator++ ( )

Operador ++ para el preincremento.

5.4.3.5. const_description_iterator conjunto::const_description_iterator::operator-- (int)

Operador – para el postdecremento del iterador.

5.4.3.6. const_description_iterator& conjunto::const_description_iterator::operator-- ( )

Operador – para el predecremento del iterador.

5.4.3.7. const_description_iterator& conjunto::const_description_iterator::operator= (const const_description_iterator & *it*)

Operador de asignacion.

Parámetros

in	*it*	iterador a asignar

5.4.3.8. bool conjunto::const_description_iterator::operator== (const const_description_iterator & *it*)

Operador == que comprueba si dos iteradores son iguales.

Parámetros

in	*it*	iterador a comparar

5.4.4. Documentación de las funciones relacionadas y clases amigas

5.4.4.1. friend class conjunto [friend]

5.4.5. Documentación de los datos miembro

5.4.5.1. vector<conjunto::entrada>::const_iterator conjunto::const_description_iterator::c_itv [private]

5.4.5.2. string conjunto::const_description_iterator::descr [private]

5.4.5.3. const vector<entrada>* conjunto::const_description_iterator::mi_conj [private]

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

- conjunto.h

5.5. Referencia de la Clase conjunto::const_iterator

class const_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator*, operator++, operator++(int) operator=, operator==, operator!=

```
#include <conjunto.h>
```

Métodos públicos

- const_iterator ()

  Constructor por defecto del iterador.
- const_iterator (const const_iterator &it)

  Constructor de copia.
- const_iterator (const iterator &it)

  Convierte iterator en const_iterator.
- const conjunto::entrada & operator* () const

  *Operador * para acceder al contenido del iterador.*

- const_iterator operator++ (int)

  Operador ++ para el postincremento del iterador.
- const_iterator & operator++ ()

  Operador ++ para el preincremento.
- const_iterator operator-- (int)

  Operador – para el postdecremento del iterador.
- const_iterator & operator-- ()

  Operador – para el predecremento del iterador.
- bool operator== (const const_iterator &it)

  Operador == que comprueba si dos iteradores son iguales.
- bool operator!= (const const_iterator &it)

  Operador != que comprueba si dos iteradores son distintos.
- const_iterator & operator= (const const_iterator &it)

  Operador de asignacion.

Atributos privados

- vector< entrada >::const_iterator c_itv

Amigas

- class conjunto

5.5.1. Descripción detallada

class const_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator*, operator++, operator++(int) operator=, operator==, operator!=

5.5.2. Documentación del constructor y destructor

5.5.2.1. conjunto::const_iterator::const_iterator ( )

Constructor por defecto del iterador.

5.5.2.2. conjunto::const_iterator::const_iterator (const const_iterator & *it*)

Constructor de copia.

Parámetros

in	*it*	

5.5.2.3. conjunto::const_iterator::const_iterator (const iterator & *it*)

Convierte iterator en const_iterator.

5.5.3. Documentación de las funciones miembro

5.5.3.1. bool conjunto::const_iterator::operator!= (const const_iterator & *it*)

Operador != que comprueba si dos iteradores son distintos.

Parámetros

in	*it*	iterador a comparar

5.5.3.2. const conjunto::entrada& conjunto::const_iterator::operator∗ ( ) const

Operador ∗ para acceder al contenido del iterador.

5.5.3.3. const_iterator conjunto::const_iterator::operator++ (int)

Operador ++ para el postincremento del iterador.

5.5.3.4. const_iterator& conjunto::const_iterator::operator++ ( )

Operador ++ para el preincremento.

5.5.3.5. const_iterator conjunto::const_iterator::operator-- (int)

Operador – para el postdecremento del iterador.

5.5.3.6. const_iterator& conjunto::const_iterator::operator-- ( )

Operador – para el predecremento del iterador.

5.5.3.7. const_iterator& conjunto::const_iterator::operator= (const const_iterator & *it*)

Operador de asignacion.

Parámetros

in	*it*	iterador a asignar

5.5.3.8. bool conjunto::const_iterator::operator== (const const_iterator & *it*)

Operador == que comprueba si dos iteradores son iguales.

Parámetros

in	*it*	iterador a comparar

5.5.4. Documentación de las funciones relacionadas y clases amigas

5.5.4.1. friend class conjunto [friend]

5.5.5. Documentación de los datos miembro

5.5.5.1. vector<entrada>::const_iterator conjunto::const_iterator::c_itv [private]

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

- conjunto.h

5.6. Referencia de la Clase crimen

Clase crimen, asociada a la definición de un crimen.

`#include <crimen.h>`

Métodos públicos

- crimen ()

  Constructor primitivo.
- crimen (const crimen &x)

  Constructor de copia.
- void setID (long int &id)

  Establece la ID del crimen.
- void setCaseNumber (const string &s)

  Establece el número de caso.
- void setDate (const fecha &d)

  Establece la fecha del crimen.
- void setIUCR (const string &IU)

  Establece la IUCR (Illinois Uniform Crime Reporting) del crimen.
- void setPrimaryType (const string &PType)

  Establece tipo de delito.
- void setDescription (const string &Desc)

  Establece la descripción del crimen.
- void setLocationDescription (const string &LDesc)

  Establece la descripción del escenario del crimen.
- void setLatitude (const double &lat)

  Establece la latitud en la que se produjo el crimen.
- void setLongitude (const double &longt)

  Establece la longitud en la que se produjo el crimen.
- void setArrest (const bool a)

  Establece si se produjo un arresto.
- void setDomestic (const bool d)

  Establece si fue crimen doméstico o no.
- long int getID () const

  Devuelve la ID del crimen.
- string getCaseNumber () const

  Devuelve el número del caso.
- string getIUCR () const

  Devuelve la IUCR del crimen.
- string getDescription () const

  Devuelve la descripción del crimen.
- string getPrimaryType () const

  Devuelve el tipo de delito.
- string getLocationDescription () const

  Devuelve la descripción del escenario del crimen.
- bool getArrest () const

  Devuelve si se produjo un arresto.
- bool getDomestic () const

  Devuelve si fue un crimen doméstico.
- double getLatitude () const

  Devuelve la latitud en la que se produjo el crimen.
- double getLongitude () const

  Devuelve la longitud en la que se produjo el crimen.
- fecha getDate () const

  Devuelve la fecha del crimen.
- crimen & operator= (const crimen &c)

Sobrecarga del operador de asignación.

- bool operator== (const crimen &x) const

  Sobrecarga del operador ==.

- bool operator< (const crimen &x) const

  Sobrecarga del operador <.

Atributos privados

- long int ID
- string CaseNumber
- fecha Date
- string IUCR
- string PrimaryType
- string Description
- string LocationDescription
- bool Arrest
- bool Domestic
- double Latitude
- double Longitude

Amigas

- ostream & operator<< (ostream &, const crimen &)

  Sobrecarga de la salida estándar.

5.6.1. Descripción detallada

Clase crimen, asociada a la definición de un crimen.

crimen::crimen, ...

! Métodos –> crimen::crimen(), crimen::crimen(const crimen& x), setID(long int & id), setCaseNumber(const string & s), setDate(const fecha & d), setIUCR(const string &IU), setPrimaryType(const string &PType), setDescription(const string &Desc), setLocationDescription(const string &LDesc), setLatitude(const double &lat), setLongitude(const double &longt), setArrest(const bool a), setDomestic(const bool d), getID( ), getCaseNumber( ), getIUCR(), getDescription(), getPrimaryType(), getLocationDescription(), getArrest(), getDomestic(), getLatitude(), getLongitude(), getDate( )

Descripción Contiene toda la información asociada a un crimen.

5.6.2. Documentación del constructor y destructor

5.6.2.1. crimen::crimen ( )

Constructor primitivo.

5.6.2.2. crimen::crimen (const crimen & *x*)

Constructor de copia.

Parámetros

in	*x*	objeto crimen a copiar

5.6.3. Documentación de las funciones miembro

5.6.3.1. bool crimen::getArrest ( ) const

Devuelve si se produjo un arresto.

Devuelve

true si hubo arresto, false en caso contrario

5.6.3.2. string crimen::getCaseNumber ( ) const

Devuelve el número del caso.

5.6.3.3. fecha crimen::getDate ( ) const

Devuelve la fecha del crimen.

Devuelve

devuelve un objeto de la clase Fecha

5.6.3.4. string crimen::getDescription ( ) const

Devuelve la descripción del crimen.

5.6.3.5. bool crimen::getDomestic ( ) const

Devuelve si fue un crimen doméstico.

Devuelve

true si fue doméstico, false en caso contrario

5.6.3.6. long int crimen::getID ( ) const

Devuelve la ID del crimen.

5.6.3.7. string crimen::getIUCR ( ) const

Devuelve la IUCR del crimen.

5.6.3.8. double crimen::getLatitude ( ) const

Devuelve la latitud en la que se produjo el crimen.

5.6.3.9. string crimen::getLocationDescription ( ) const

Devuelve la descripción del escenario del crimen.

5.6.3.10. double crimen::getLongitude ( ) const

Devuelve la longitud en la que se produjo el crimen.

5.6.3.11. string crimen::getPrimaryType ( ) const

Devuelve el tipo de delito.

5.6.3.12. bool crimen::operator< (const crimen & *x*) const

Sobrecarga del operador <.

Devuelve

Devuelve true si la ID es menor.

5.6.3.13. crimen& crimen::operator= (const crimen & *c*)

Sobrecarga del operador de asignación.

5.6.3.14. bool crimen::operator== (const crimen & *x*) const

Sobrecarga del operador ==.

Devuelve

Devuelve true si la ID de dos casos es la misma, false en caso contrario.

5.6.3.15. void crimen::setArrest (const bool *a*)

Establece si se produjo un arresto.

Parámetros

in	*a*	true si se produjo, false en caso contrario

5.6.3.16. void crimen::setCaseNumber (const string & *s*)

Establece el número de caso.

Parámetros

in	*s*	Número del caso, formato string

5.6.3.17. void crimen::setDate (const fecha & *d*)

Establece la fecha del crimen.

Parámetros

in	*d*	objeto de la clase Fecha

5.6.3.18. void crimen::setDescription (const string & *Desc*)

Establece la descripción del crimen.

Parámetros

in	*Desc*	Descripción en formato string

5.6.3.19. void crimen::setDomestic (const bool *d*)

Establece si fue crimen doméstico o no.

Parámetros

in	*d*	true si lo fue, false en caso contrario

5.6.3.20. void crimen::setID (long int & *id*)

Establece la ID del crimen.

Parámetros

in	*id*	ID

5.6.3.21. void crimen::setIUCR (const string & *IU*)

Establece la IUCR (Illinois Uniform Crime Reporting) del crimen.

Parámetros

in	*IU*	IUCR,formato string

5.6.3.22. void crimen::setLatitude (const double & *lat*)

Establece la latitud en la que se produjo el crimen.

Parámetros

in	*lat*	Latitud

5.6.3.23. void crimen::setLocationDescription (const string & *LDesc*)

Establece la descripción del escenario del crimen.

Parámetros

in	*LDesc*	Descripcion del escenario, formato string

5.6.3.24. void crimen::setLongitude (const double & *longt*)

Establece la longitud en la que se produjo el crimen.

Parámetros

in	*longt*	Longitud

5.6.3.25. void crimen::setPrimaryType (const string & *PType*)

Establece tipo de delito.

Parámetros

in	*PType*	Tipo de delito, formato string

5.6.4. Documentación de las funciones relacionadas y clases amigas

5.6.4.1. ostream& operator<< (ostream & , const crimen &) [friend]

Sobrecarga de la salida estándar.

Devuelve

Escribe cada uno de los atributos junto a su valor en una línea distinta.

5.6.5. Documentación de los datos miembro

5.6.5.1. bool crimen::Arrest [private]

5.6.5.2. string crimen::CaseNumber [private]

5.6.5.3. fecha crimen::Date [private]

5.6.5.4. string crimen::Description [private]

5.6.5.5. bool crimen::Domestic [private]

5.6.5.6. long int crimen::ID [private]

5.6.5.7. string crimen::IUCR [private]

5.6.5.8. double crimen::Latitude [private]

5.6.5.9. string crimen::LocationDescription [private]

5.6.5.10. double crimen::Longitude [private]

5.6.5.11. string crimen::PrimaryType [private]

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

- crimen.h

5.7. Referencia de la Clase conjunto::description_iterator

class description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion

`#include <conjunto.h>`

Métodos públicos

- description_iterator ()

  Constructor por defecto del iterador.
- description_iterator (const description_iterator &it)

  Constructor de copia.
- const conjunto::entrada & operator∗ () const

  Operador ∗ para acceder al contenido del iterador.
- description_iterator operator++ (int)

  Operador ++ para el postincremento del iterador.
- description_iterator & operator++ ()

  Operador ++ para el preincremento.
- description_iterator operator-- (int)

  Operador – para el postdecremento del iterador.
- description_iterator & operator-- ()

  Operador – para el predecremento del iterador.
- bool operator== (const description_iterator &it)

  Operador == que comprueba si dos iteradores son iguales.
- bool operator!= (const description_iterator &it)

  Operador != que comprueba si dos iteradores son distintos.
- description_iterator & operator= (const description_iterator &it)

  Operador de asignacion.

Atributos privados

- const vector< entrada > * mi_conj
- string descr
- vector< entrada >::iterator itv

Amigas

- class conjunto

5.7.1. Descripción detallada

class description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator*, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion

5.7.2. Documentación del constructor y destructor

5.7.2.1. conjunto::description_iterator::description_iterator ( )

Constructor por defecto del iterador.

5.7.2.2. conjunto::description_iterator::description_iterator (const description_iterator & *it*)

Constructor de copia.

Parámetros

in	*it*	

5.7.3. Documentación de las funciones miembro

5.7.3.1. bool conjunto::description_iterator::operator!= (const description_iterator & *it*)

Operador != que comprueba si dos iteradores son distintos.

Parámetros

in	*it*	iterador a comparar

5.7.3.2. const conjunto::entrada& conjunto::description_iterator::operator* ( ) const

Operador * para acceder al contenido del iterador.

5.7.3.3. description_iterator conjunto::description_iterator::operator++ (int)

Operador ++ para el postincremento del iterador.

5.7.3.4. description_iterator& conjunto::description_iterator::operator++ ( )

Operador ++ para el preincremento.

5.7.3.5. description_iterator conjunto::description_iterator::operator-- (int)

Operador – para el postdecremento del iterador.

5.7.3.6. description_iterator& conjunto::description_iterator::operator-- ( )

Operador – para el predecremento del iterador.

5.7.3.7. description_iterator& conjunto::description_iterator::operator= (const description_iterator & *it*)

Operador de asignacion.

Parámetros

in	*it*	iterador a asignar

5.7.3.8. bool conjunto::description_iterator::operator== (const description_iterator & *it*)

Operador == que comprueba si dos iteradores son iguales.

Parámetros

in	*it*	iterador a comparar

5.7.4. Documentación de las funciones relacionadas y clases amigas

5.7.4.1. friend class conjunto [friend]

5.7.5. Documentación de los datos miembro

5.7.5.1. string conjunto::description_iterator::descr [private]

5.7.5.2. vector<entrada>::iterator conjunto::description_iterator::itv [private]

5.7.5.3. const vector<entrada>* conjunto::description_iterator::mi_conj [private]

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

- conjunto.h

5.8. Referencia de la Clase fecha

Clase fecha, asociada a la.

`#include <fecha.h>`

Métodos públicos

- fecha ()
- fecha (const string &s)
- fecha (const fecha &x)
- fecha & operator= (const fecha &f)
- fecha & operator= (const string &s)
- string toString () const
- bool operator== (const fecha &f) const
- bool operator< (const fecha &f) const
- bool operator> (const fecha &f) const
- bool operator<= (const fecha &f) const
- bool operator>= (const fecha &f) const
- bool operator!= (const fecha &f) const

=

Atributos privados

- int sec
- int min
- int hour
- int mday
- int mon
- int year

Amigas

- ostream & operator<< (ostream &os, const fecha &f)

5.8.1. Descripción detallada

Clase fecha, asociada a la.

fecha::fecha, ... Descripción contiene toda la información asociada a una fecha con el formato mm/dd/aaaa hh-:mm:ss AM/PM

Tareas pendientes Escribe la documentación de la clase

Implementar esta clase

5.8.2. Documentación del constructor y destructor

5.8.2.1. fecha::fecha ( )

5.8.2.2. fecha::fecha (const string & *s*)

5.8.2.3. fecha::fecha (const fecha & *x*)

5.8.3. Documentación de las funciones miembro

5.8.3.1. bool fecha::operator!= (const fecha & *f*) const

=

5.8.3.2. bool fecha::operator< (const fecha & *f*) const

5.8.3.3. bool fecha::operator<= (const fecha & *f*) const

5.8.3.4. fecha & fecha::operator= (const fecha & *f*)

5.8.3.5. fecha & fecha::operator= (const string & *s*)

5.8.3.6. bool fecha::operator== (const fecha & *f*) const

5.8.3.7. bool fecha::operator> (const fecha & *f*) const

5.8.3.8. bool fecha::operator>= (const fecha & *f*) const

5.8.3.9. string fecha::toString ( ) const

5.8.4. Documentación de las funciones relacionadas y clases amigas

5.8.4.1. ostream& operator<< (ostream & *os,* const fecha & *f*) [friend]

5.8.5. Documentación de los datos miembro

5.8.5.1. int fecha::hour [private]

5.8.5.2. int fecha::mday [private]

5.8.5.3. int fecha::min [private]

5.8.5.4. int fecha::mon [private]

5.8.5.5. int fecha::sec [private]

5.8.5.6. int fecha::year [private]

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

- fecha.h
- fecha.hxx

5.9. Referencia de la Clase conjunto::iterator

class iterator forward iterador sobre el conjunto, LECTURA iterator() ,operator∗(), operator++, operator++(int) operator=, operator==, operator!=

`#include <conjunto.h>`

Métodos públicos

- iterator ()

  Constructor por defecto del iterador.
- iterator (const iterator &it)

  Constructor de copia.
- const entrada & operator∗ () const

  Operador ∗ para acceder al contenido del iterador.
- iterator operator++ (int)

  Operador ++ para el postincremento del iterador.
- iterator & operator++ ()

  Operador ++ para el preincremento.
- iterator operator-- (int)

  Operador – para el postdecremento del iterador.
- iterator & operator-- ()

  Operador – para el predecremento del iterador.
- bool operator== (const iterator &it)

  Operador == que comprueba si dos iteradores son iguales.
- bool operator!= (const iterator &it)

  Operador != que comprueba si dos iteradores son distintos.
- iterator & operator= (const iterator &it)

  Operador de asignacion.

Atributos privados

- vector< entrada >::iterator itv

Amigas

- class conjunto

5.9.1. Descripción detallada

class iterator forward iterador sobre el conjunto, LECTURA iterator() ,operator∗(), operator++, operator++(int) operator=, operator==, operator!=

5.9.2. Documentación del constructor y destructor

5.9.2.1. conjunto::iterator::iterator ( )

Constructor por defecto del iterador.

5.9.2.2. conjunto::iterator::iterator (const iterator & *it*)

Constructor de copia.

Parámetros

in	*it*	

5.9.3. Documentación de las funciones miembro

5.9.3.1. bool conjunto::iterator::operator!= (const iterator & *it*)

Operador != que comprueba si dos iteradores son distintos.

Parámetros

in	*it*	iterador a comparar

5.9.3.2. const entrada& conjunto::iterator::operator∗ ( ) const

Operador ∗ para acceder al contenido del iterador.

5.9.3.3. iterator conjunto::iterator::operator++ (int)

Operador ++ para el postincremento del iterador.

5.9.3.4. iterator& conjunto::iterator::operator++ ( )

Operador ++ para el preincremento.

5.9.3.5. iterator conjunto::iterator::operator-- (int)

Operador – para el postdecremento del iterador.

5.9.3.6. iterator& conjunto::iterator::operator-- ( )

Operador – para el predecremento del iterador.

5.9.3.7. iterator& conjunto::iterator::operator= (const iterator & *it*)

Operador de asignacion.

Parámetros

in	*it*	iterador a asignar

5.9.3.8. bool conjunto::iterator::operator== (const iterator & *it*)

Operador == que comprueba si dos iteradores son iguales.

Parámetros

in	*it*	iterador a comparar

5.9.4. Documentación de las funciones relacionadas y clases amigas

5.9.4.1. friend class conjunto [friend]

5.9.5. Documentación de los datos miembro

5.9.5.1. vector<entrada>::iterator conjunto::iterator::itv [private]

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

- conjunto.h

6. Documentación de archivos

6.1. Referencia del Archivo conjunto.h

```
#include <string>
#include <vector>
#include <iostream>
#include "crimen.h"
#include "conjunto.hxx"
```

Clases

- class conjunto

  Clase conjunto.
- class conjunto::iterator

  class iterator forward iterador sobre el conjunto, LECTURA iterator() ,operator∗(), operator++, operator++(int) operator=, operator==, operator!=
- class conjunto::const_iterator

  class const_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!=
- class conjunto::description_iterator

  class description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion
- class conjunto::const_description_iterator

  class const_description_iterator forward iterador constante sobre el diccionario, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!= esta clase itera sobre todos los elementos que emparejan con una descripcion
- class conjunto::arrest_iterator

class arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!=

- class conjunto::const_arrest_iterator

  class const_arrest_iterator forward iterador constante sobre el conjunto, Lectura const_iterator ,operator∗, operator++, operator++(int) operator=, operator==, operator!=

Funciones

- ostream & operator<< (ostream &sal, const conjunto &D)

  imprime todas las entradas del conjunto

6.1.1. Documentación de las funciones

6.1.1.1. ostream& operator<< (ostream & *sal,* const conjunto & *D*)

imprime todas las entradas del conjunto

Postcondición

No se modifica el conjunto.

Tareas pendientes implementar esta funcion

6.2. Referencia del Archivo crimen.h

```
#include <string>
#include <iostream>
#include "fecha.h"
#include "crimen.hxx"
```

Clases

- class crimen

  Clase crimen, asociada a la definición de un crimen.

Funciones

- ostream & operator<< (ostream &, const crimen &)

6.2.1. Documentación de las funciones

6.2.1.1. ostream& operator<< (ostream & , const crimen &)

Devuelve

Escribe cada uno de los atributos junto a su valor en una línea distinta.

6.3. Referencia del Archivo documentacion.dox

6.4. Referencia del Archivo fecha.h

```
#include <string>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include "fecha.hxx"
```

Clases

- class fecha

  Clase fecha, asociada a la.

Funciones

- ostream & operator<< (ostream &os, const fecha &f)

6.4.1. Documentación de las funciones

6.4.1.1. ostream& operator<< (ostream & *os,* const fecha & *f*)

6.5. Referencia del Archivo fecha.hxx

Funciones

- ostream & operator<< (ostream &os, const fecha &f)

6.5.1. Documentación de las funciones

6.5.1.1. ostream& operator<< (ostream & *os,* const fecha & *f*)

6.6. Referencia del Archivo principal.cpp

```
#include "conjunto.h"
#include "fecha.h"
#include <iostream>
#include <fstream>
```

Funciones

- bool to_bool (const string &str)

  Pasa a tipo de dato booleano el string.
- bool load (conjunto &C, const string &s)

  lee un fichero de delitos, linea a linea
- int main ()

6.6.1. Documentación de las funciones

6.6.1.1. bool load (conjunto & *C,* const string & *s*)

lee un fichero de delitos, linea a linea

Parámetros

in	*s*	nombre del fichero
in,out	*C*	conjunto sobre el que se lee

Devuelve

true si la lectura ha sido correcta, false en caso contrario

6.6.1.2. int main ( )

6.6.1.3. bool to_bool (const string & *str*)

Pasa a tipo de dato booleano el string.

Parámetros

in	*str*	cadena a convertir

Devuelve

true si la cadena es "true", false en caso contrario

Índice alfabético