PRACTICA 4 ED - FUNCTORES

Autor:

Cristian Vélez Ruiz

Version 1.0

12/19/2015

Autor:

Cristian Vélez Ruiz

Version 1.0

19/12/2015

Table of Contents

Table of contents

Todo List

Class conjunto< CMP >

Implementa esta clase, junto con su documentación asociada

Class fecha

Implementa esta clase, junto con su documentación asociada

Member operator<< (ostream &sal, const conjunto< CMP > &D)

implementar esta funcion

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Class Documentation

conjunto< CMP > Class Template Reference

Clase conjunto.

`#include <conjunto.h>`

Classes

class **const_iterator**

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

class **iterator**

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

typedef **crimen entrada**
typedef unsigned int **size_type**

Public Member Functions

conjunto ()

constructor primitivo.

conjunto (typename **conjunto**< CMP >::**iterator** &inicio, typename **conjunto**< CMP >::**iterator** &fina)

Constructor que se le pasan dos iteradores ,.

conjunto (const **conjunto** &d)

constructor de copia

conjunto< CMP >::**iterator find** (const **crimen** &c)

Metodo para buscar un crimen , que te devuelve un iterador a su posicion.

conjunto< CMP >::**const_iterator find** (const **crimen** &c) const

Metodo para buscar un crimen , que te devuelve un const_iterador a su posicion.

conjunto< CMP >::**iterator find** (const long int &id)

Metodo para buscar un crimen respecto a su ID , que te devuelve un iterador a su posicion.

conjunto< CMP >::**const_iterator find** (const long int &id) const

Metodo para buscar un crimen respecto a su ID , que te devuelve un const_iterador a su posicion.

conjunto< CMP >::**iterator lower_bound** (const **conjunto**< CMP >::**entrada** &x)

Devuelve un iterador al elemento que no precede a x , en resumen , devuelve el primer elemento comp(e,x) es falso.

conjunto< CMP >::**const_iterator lower_bound** (const **conjunto**< CMP >::**entrada** &x) const

Devuelve un const_iterador al elemento que no precede a x , en resumen , devuelve el primer elemento comp(e,x) es falso.

conjunto< CMP >::**iterator upper_bound** (const **conjunto**< CMP >::**entrada** &x)

Devuelve un iterador al elemento que precede a x , en resumen , devuelve el primer elemento comp(e,x) es verdadero.

conjunto< CMP >::**const_iterator upper_bound** (const **conjunto**< CMP >::**entrada** &x) const

Devuelve un const_iterador al elemento que precede a x , en resumen , devuelve el primer elemento comp(e,x) es verdadero.

pair< **conjunto::entrada**, bool > **findID** (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

int **busquedaBinaria** (const **crimen** &e) const
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 una entrada en el conjunto .

bool **erase** (const **conjunto::entrada** &e)
conjunto & **operator=** (const **conjunto** &org)

operador de asignación

size_type size () const

numero de entradas en el conjunto

bool **empty** () const

*vacia Chequea si el conjunto esta vacio (**size()**==0)*

vector< **crimen** > **getVector** ()

Consultor Vector Devuelve el valor del vector vc.

conjunto::entrada getPos (unsigned int indice) const
iterator begin ()

devuelve iterador al inicio del conjunto

iterator end ()

devuelve iterador al final (posición siguiente al último del conjunto

const_iterator cbegin () const
const_iterator cend () const

iterador al final

Friends

class **iterator**
class **arrest_iterator**
template<typename UMP > ostream & **operator<<** (ostream &sal, const **conjunto**< CMP > &D)

imprime todas las entradas del conjunto

Detailed Description

template<typename CMP>class conjunto< CMP >

Clase conjunto.

conjunto:: conjunto, find, size, Tipos **conjunto::entrada**, **conjunto::size_type** Descripción

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 métodos (inserción de elementos, consulta de un elemento, etc). Este conjunto "simulará" un conjunto de la stl, con algunas claras diferencias pues, entre otros, no estará dotado de la capacidad de iterar (recorrer) a través 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=

El número de elementos en el conjunto puede variar dinámicamente; la gestión de la memoria es automática.

Todo:

Implementa esta clase, junto con su documentación asociada

Member Typedef Documentation

template<typename CMP> typedef crimen conjunto< CMP >::entrada

template<typename CMP> typedef unsigned int conjunto< CMP >::size_type

---

Constructor & Destructor Documentation

template<typename CMP > conjunto< CMP >::conjunto ()

constructor primitivo.

Postcondition:

define la entrada nula como el par ("",-1)

template<typename CMP> conjunto< CMP >::conjunto (typename conjunto< CMP >::iterator & *inicio*, typename conjunto< CMP >::iterator & *fina*)

Constructor que se le pasan dos iteradores ,.

Parameters:

inicio,***conjunto<CMP>::iterator***	
fina,***conjunto<CMP>::iterator***	

template<typename CMP> conjunto< CMP >::conjunto (const conjunto< CMP > & *d*)

constructor de copia

Parameters:

in	*d*	conjunto a copiar

Member Function Documentation

template<typename CMP > conjunto< CMP >::iterator conjunto< CMP >::begin ()

devuelve iterador al inicio del conjunto

template<typename CMP > int conjunto< CMP >::busquedaBinaria (const crimen & *e*) const

template<typename CMP > conjunto< CMP >::const_iterator conjunto< CMP >::cbegin () const

Returns:

Devuelve el **const_iterator** a la primera posición del conjunto.

Postcondition:

no modifica el diccionario

template<typename CMP > conjunto< CMP >::const_iterator conjunto< CMP >::cend () const

iterador al final

Returns:

Devuelve el iterador constante a la posición final del conjunto.

Postcondition:

no modifica el diccionario

template<typename CMP > bool conjunto< CMP >::empty () const

vacia Chequea si el conjunto esta vacio (**size()**==0)

template<typename CMP > conjunto< CMP >::iterator conjunto< CMP >::end ()

devuelve iterador al final (posición siguiente al último del conjunto

template<typename CMP > bool conjunto< CMP >::erase (const long int & *id*)

Borra una entrada en el conjunto .

Busca la entrada con id en el conjunto (o e.getID() en el segundo caso) y si la encuentra la borra

Parameters:

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

Postcondition:

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

template<typename CMP> bool conjunto< CMP >::erase (const conjunto< CMP >::entrada & *e*)

template<typename CMP > conjunto< CMP >::iterator conjunto< CMP >::find (const crimen & *c*)

Metodo para buscar un crimen , que te devuelve un iterador a su posicion.

Parameters:

c,tipo	crimen

Returns:

conjunto<CMP>::iterator, iterador a su posicion

template<typename CMP > conjunto< CMP >::const_iterator conjunto< CMP >::find (const crimen & *c*) const

Metodo para buscar un crimen , que te devuelve un const_iterador a su posicion.

Parameters:

c,tipo	crimen

Returns:

conjunto<CMP>::const_iterator, iterador a su posicion

template<typename CMP > conjunto< CMP >::iterator conjunto< CMP >::find (const long int & *id*)

Metodo para buscar un crimen respecto a su ID , que te devuelve un iterador a su posicion.

Parameters:

c,tipo	crimen

Returns:

conjunto<CMP>::iterator, iterador a su posicion

template<typename CMP > conjunto< CMP >::const_iterator conjunto< CMP >::find (const long int & *id*) const

Metodo para buscar un crimen respecto a su ID , que te devuelve un const_iterador a su posicion.

Parameters:

c,tipo	crimen

Returns:

conjunto<CMP>::const_iterator, iterador a su posicion

template<typename CMP > conjunto< CMP > conjunto< CMP >::findDESCR (const string & *descr*) const

busca los crimenes que contienen una determinada descripcion

Parameters:

descr	string que representa la descripcion del delito buscar

Returns:

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

Postcondition:

no modifica el conjunto.

Uso

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

template<typename CMP > pair< typename conjunto< CMP >::entrada, bool > conjunto< CMP >::findID (const long int & *id*) const

busca un crimen en el conjunto

Parameters:

id	identificador del crimen buscar

Returns:

Si existe una entrada en el conjunto devuelve un par con una copia de la entrada en el conjunto y con el segundo valor a true. Si no se encuentra devuelve la entrada con la definicion por defecto y false

Postcondition:

no modifica el conjunto.
Uso

```
if (C.findID(12345).second ==true) cout << "Esta" ;
else cout << "No esta";
```

template<typename CMP > conjunto< CMP > conjunto< CMP >::findIUCR (const string & *iucr*) const

busca los crimenes con el mismo codigo IUCR

Parameters:

icur	identificador del crimen buscar

Returns:

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

Postcondition:

no modifica el conjunto.
Uso

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

template<typename CMP> conjunto::entrada conjunto< CMP >::getPos (unsigned int *indice*) const[inline]

template<typename CMP> vector<crimen> conjunto< CMP >::getVector ()[inline]

Consultor Vector Devuelve el valor del vector vc.

template<typename CMP> bool conjunto< CMP >::insert (const conjunto< CMP >::entrada & *e*)

Inserta una entrada en el conjunto.

Parameters:

e	entrada a insertar

Returns:

true si la entrada se ha podido insertar con éxito, esto es, no existe un delito con igual ID en el conjunto. False en caso contrario

Postcondition:

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

template<class CMP> conjunto< CMP >::iterator conjunto< CMP >::lower_bound (const conjunto< CMP >::entrada & *x*)

Devuelve un iterador al elemento que no precede a x , en resumen , devuelve el primer elemento comp(e,x) es falso.

Parameters:

entrada,tipo	**conjunto::entrada**

Returns:

conjunto<CMP>::iterator, iterador a su posicion

template<class CMP> conjunto< CMP >::const_iterator conjunto< CMP >::lower_bound (const conjunto< CMP >::entrada & *x*) const

Devuelve un const_iterador al elemento que no precede a x , en resumen , devuelve el primer elemento comp(e,x) es falso.

Parameters:

entrada,tipo	**conjunto::entrada**

Returns:

conjunto<CMP>::const_iterator, iterador a su posicion

template<typename CMP > conjunto< CMP > & conjunto< CMP >::operator= (const conjunto< CMP > & *org*)

operador de asignación

Parameters:

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

template<typename CMP > conjunto< CMP >::size_type conjunto< CMP >::size () const

numero de entradas en el conjunto

Postcondition:

No se modifica el conjunto.

template<class CMP> conjunto< CMP >::iterator conjunto< CMP >::upper_bound (const conjunto< CMP >::entrada & *x*)

Devuelve un iterador al elemento que precede a x , en resumen , devuelve el primer elemento comp(e,x) es verdadero.

Parameters:

entrada,tipo	**conjunto::entrada**

Returns:

conjunto<CMP>::iterator, iterador a su posicion

template<class CMP> conjunto< CMP >::const_iterator conjunto< CMP >::upper_bound (const conjunto< CMP >::entrada & *x*) const

Devuelve un const_iterador al elemento que precede a x , en resumen , devuelve el primer elemento comp(e,x) es verdadero.

Parameters:

entrada,tipo	**conjunto::entrada**

Returns:

conjunto<CMP>::const_iterator, iterador a su posicion

Friends And Related Function Documentation

template<typename CMP> friend class arrest_iterator`[friend]`

template<typename CMP> friend class iterator`[friend]`

template<typename CMP> template<typename UMP > ostream& operator<< (ostream & *sal*, const conjunto< CMP > & *D*)`[friend]`

imprime todas las entradas del conjunto

Postcondition:

No se modifica el conjunto.

Todo:

implementar esta funcion

The documentation for this class was generated from the following files:

1. include/**conjunto.h**
2. src/**conjunto.hxx**

conjunto< CMP >::const_iterator Class Reference

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

`#include <conjunto.h>`

Public Member Functions

const_iterator ()
*Constructor por defecto de **const_iterator**.*

const_iterator (const **const_iterator** &it)
Constructor de Copia.

const_iterator (const vector< **conjunto::entrada** >::**const_iterator** n)
Constructor con parametro iterator tipo vecto<entrada>::const_iterator.

const_iterator (const **iterator** &it)
*Convierte iterator en **const_iterator**.*

const **conjunto::entrada** & **operator*** () const
Devuelve el valor del iterador actual.

const_iterator operator++ (int)
Pre incremento ++ , ++it.

const_iterator & **operator++** ()
Post incremento ++ , it++.

const_iterator operator-- (int)
Pre decremento - , -it.

const_iterator & **operator--** ()
Post decremento - , it-.

bool **operator==** (const **const_iterator** &it)
*Sobrecarga operator == de clase **conjunto::const_iterator**.*

bool **operator!=** (const **const_iterator** &it)
*Sobrecarga operator == de clase **conjunto::const_iterator**.*

Friends

class **conjunto**

Detailed Description

template<typename CMP>class conjunto< CMP >::const_iterator

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

Constructor & Destructor Documentation

template<typename CMP> conjunto< CMP >::const_iterator::const_iterator ()

Constructor por defecto de **const_iterator**.

template<typename CMP> conjunto< CMP >::const_iterator::const_iterator (const const_iterator & *it*)

Constructor de Copia.

Parameters:

it,tipo	**const_iterator**

template<typename CMP> conjunto< CMP >::const_iterator::const_iterator (const vector< conjunto::entrada >::const_iterator *n*)

Constructor con parametro iterator tipo vecto<entrada>::const_iterator.

Parameters:

*n,**const_iterator***	tipo vecto<entrada>

template<typename CMP> conjunto< CMP >::const_iterator::const_iterator (const iterator & *it*)

Convierte iterator en **const_iterator**.

Member Function Documentation

template<typename CMP> bool conjunto< CMP >::const_iterator::operator!= (const const_iterator & *it*)

Sobrecarga operator == de clase **conjunto::const_iterator**.

Returns:

false, si son iguales , true , si son distintos

template<typename CMP> const conjunto< CMP >::entrada & conjunto< CMP >::const_iterator::operator* () const

Devuelve el valor del iterador actual.

Returns:

conjunto::entrada , valor actual del iterator

template<typename CMP> conjunto< CMP >::const_iterator conjunto< CMP >::const_iterator::operator++ (int)

Pre incremento ++ , ++it.

Returns:

Devuelve el const_iterador

Postcondition:

Incrementa el iterador despues de devolverlo

template<typename CMP> conjunto< CMP >::const_iterator & conjunto< CMP >::const_iterator::operator++ ()

Post incremento ++ , it++.

Returns:

Devuelve el const_iterador

Postcondition:

Incrementa el iterador para devolverlo

template<typename CMP> conjunto< CMP >::const_iterator conjunto< CMP >::const_iterator::operator-- (int)

Pre decremento - , -it.

Returns:

Devuelve el const_iterador

Postcondition:

Decrementa el iterador despues de devolverlo

template<typename CMP> conjunto< CMP >::const_iterator & conjunto< CMP >::const_iterator::operator-- ()

Post decremento - , it-.

Returns:

Devuelve el const_iterador decrementado

Postcondition:

Decrementa iterator

template<typename CMP> bool conjunto< CMP >::const_iterator::operator== (const const_iterator & *it*)

Sobrecarga operator == de clase **conjunto::const_iterator**.

Returns:

true , si son iguales , false , si son distintos

Friends And Related Function Documentation

template<typename CMP> friend class conjunto [friend]

The documentation for this class was generated from the following files:

3 include/**conjunto.h**
4 src/**conjunto.hxx**

CrecienteFecha Class Reference

Class **CrecienteFecha** operator()

`#include <conjunto.h>`

Public Member Functions

bool **operator()** (const **crimen** &a, const **crimen** &b)

Comparador del funtor, para ordenar de forma creciente por fecha.

Detailed Description

Class **CrecienteFecha** operator()

Member Function Documentation

bool CrecienteFecha::operator() (const crimen & *a*, const crimen & *b*)

Comparador del funtor, para ordenar de forma creciente por fecha.

Parameters:

a,*tipo*	crimen
b,*tipo*	crimen

The documentation for this class was generated from the following files:

5 include/**conjunto.h**
6 src/**conjunto.hxx**

CrecienteID Class Reference

Class **CrecienteID** operator()

`#include <conjunto.h>`

Public Member Functions

bool **operator()** (const **crimen** &a, const **crimen** &b)

Comparador del funtor, para ordenar de forma creciente por ID.

Detailed Description

Class **CrecienteID** operator()

Member Function Documentation

bool CrecienteID::operator() (const crimen & *a*, const crimen & *b*)

Comparador del funtor, para ordenar de forma creciente por ID.

Parameters:

a,*tipo*	crimen
b,*tipo*	crimen

The documentation for this class was generated from the following files:

7 include/**conjunto.h**
8 src/**conjunto.hxx**

CrecienteIUCR Class Reference

Class **CrecienteIUCR** operator()

`#include <conjunto.h>`

Public Member Functions

bool **operator()** (const **crimen** &a, const **crimen** &b)

Comparador del funtor , para ordenar de forma creciente por IUCR.

Detailed Description

Class **CrecienteIUCR** operator()

Member Function Documentation

bool CrecienteIUCR::operator() (const crimen & *a*, const crimen & *b*)

Comparador del funtor , para ordenar de forma creciente por IUCR.

Parameters:

a,*tipo*	crimen
b,*tipo*	crimen

The documentation for this class was generated from the following files:

9 include/**conjunto.h**
10 src/**conjunto.hxx**

crimen Class Reference

`#include <crimen.h>`

Public Member Functions

crimen ()
Constructor por defecto de Crimen.

crimen (const **crimen** &x)
Constructor con parametro Crimen.

void **setCrimen** (const string &cadena)
Fija un Crimen con parametro String.

void **setID** (long int &id)
Fija un ID a un Crimen.

void **setCaseNumber** (const string &s)
Fija un CaseNumber a un Crimen.

void **setDate** (const **fecha** &d)
Fija una Fecha a un Crimen.

void **setIUCR** (string iucr)
Fija un IUCR a un Crimen.

void **setPrimaryType** (string primary_type)
Fija un Tipo Primario a un Crimen.

void **setDescr** (string descripcion)
Fija una Descripcion a un Crimen.

void **setLocalDescription** (string descripcion_local)
Fija una Descripcion del lugar a un Crimen.

void **setArrest** (bool a)
Fija un valor al Arresto en un Crimen.

void **setDomestic** (bool d)
Fija un valor a la variable Domestic de un Crimen.

void **setLatitude** (double latitud)
Fija un valor a la variable Latitud de un Crimen.

void **setLongitude** (double longitud)
Fija un valor a la variable Longitud de un Crimen.

long int **getID** () const
Devuelve el valor de la ID del Crimen.

string **getCaseNumber** () const
Devuelve el valor del CaseNumber de un Crimen.

fecha getDate () const
Devuelve el valor de la fecha de un Crimen.

string **getIUCR** () const
Devuelve el valor del IUCR de un Crimen.

string **getPrimaryType** () const
Devuelve el valor del PrimaryType de un Crimen.

string **getDescr** () const
Devuelve el valor de la Description de un Crimen.

string **getLocalDescription** () const
Devuelve el valor del LocalDescription de un Crimen.

bool **getArrest** () const
Devuelve el valor del Arrest de un Crimen.

bool **getDomestic** () const
Devuelve el valor del Domestic de un Crimen.

double **getLatitude** () const
Devuelve el valor de la Latitude de un Crimen.

double **getLongitude** () const
Devuelve el valor de la Longitude de un Crimen.

crimen & **operator**= (const **crimen** &c)
Sobrecarga del operador igual (=) con parametro de tipo Crimen.

bool **operator**== (const **crimen** &x) const
Sobrecarga del operador (==) con parametro tipo Crimen.

bool **operator**< (const **crimen** &x) const
Sobrecarga del operador (<) con parametro tipo Crimen.

bool **operator**<= (const **crimen** &x) const

Constructor & Destructor Documentation

crimen::crimen ()

Constructor por defecto de Crimen.

fichero de implementacion de la clase crimen

Postcondition:
inicializa todo a 0.

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

Constructor con parametro Crimen.

Parameters:

x	Crimen a convertir , para inicializar la clase

Member Function Documentation

bool crimen::getArrest () const

Devuelve el valor del Arrest de un Crimen.

Returns:
bool , Arrest

Postcondition:
no modifica la clase.

string crimen::getCaseNumber () const

Devuelve el valor del CaseNumber de un Crimen.

Returns:
string , CaseNumber

Postcondition:
no modifica la clase.

fecha crimen::getDate () const

Devuelve el valor de la fecha de un Crimen.

Returns:
fecha , Date

Postcondition:
no modifica la clase.

string crimen::getDescr () const

Devuelve el valor de la Description de un Crimen.

Returns:
string , Description

Postcondition:
no modifica la clase.

bool crimen::getDomestic () const

Devuelve el valor del Domestic de un Crimen.

Returns:

bool , Domestic

Postcondition:

no modifica la clase.

long int crimen::getID () const

Devuelve el valor de la ID del Crimen.

Returns:

int , ID

Postcondition:

no modifica la clase.

string crimen::getIUCR () const

Devuelve el valor del IUCR de un Crimen.

Returns:

string , IUCR

Postcondition:

no modifica la clase.

double crimen::getLatitude () const

Devuelve el valor de la Latitude de un Crimen.

Returns:

double , Latitude

Postcondition:

no modifica la clase.

string crimen::getLocalDescription () const

Devuelve el valor del LocalDescription de un Crimen.

Returns:

string , LocalDescription

Postcondition:

no modifica la clase.

double crimen::getLongitude () const

Devuelve el valor de la Longitude de un Crimen.

Returns:

double , Longitude

Postcondition:

no modifica la clase.

string crimen::getPrimaryType () const

Devuelve el valor del PrimaryType de un Crimen.

Returns:

string , PrimaryType

Postcondition:

no modifica la clase.

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

Sobrecarga del operador (<) con parametro tipo Crimen.

Parameters:

x	tipo de dato Crimen

Returns:

True si this es menor que x , false si es mayor

Postcondition:

no modifica la clase.

bool crimen::operator<= (const crimen & *x*) const

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

Sobrecarga del operador igual (=) con parametro de tipo Crimen.

Parameters:

c	tipo de dato Crimen

Returns:

Devuelve una refencia con la clase igual a la pasada por parametro

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

Sobrecarga del operador (==) con parametro tipo Crimen.

Parameters:

c	tipo de dato Crimen

Returns:

True si es el mismo Crimen , false si es distinta

Postcondition:

no modifica la clase.

void crimen::setArrest (bool *a*)

Fija un valor al Arresto en un Crimen.

Parameters:

a	es un Bool, que indica el valor de Arrest de un Crimen

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

Fija un CaseNumber a un Crimen.

Parameters:

s	es un String , para inicializar el valor de CaseNumber de un Crimen

void crimen::setCrimen (const string & *cadena*)

Fija un Crimen con parametro String.

Parameters:

cadena	string a convertir , para inicializar un crimen

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

Fija una Fecha a un Crimen.

Parameters:

d	es un objeto de la clase Fecha , para inicializar el valor de Date de un Crimen

void crimen::setDescr (string *descripcion*)

Fija una Descripcion a un Crimen.

Parameters:

descripcion	es un String, para inicializar el valor de DESCR de un Crimen

void crimen::setDomestic (bool *d*)

Fija un valor a la variable Domestic de un Crimen.

Parameters:

d	es un Bool, que indica el valor de Domestic de un Crimen

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

Fija un ID a un Crimen.

Parameters:

id	es un entero , para inicializar el valor de valor de ID de un crimen

void crimen::setIUCR (string *iucr*)

Fija un IUCR a un Crimen.

Parameters:

iucr	es un String, para inicializar el valor de IUCR de un Crimen

void crimen::setLatitude (double *latitud*)

Fija un valor a la variable Latitud de un Crimen.

Parameters:

latitud	es un Double, que indica el valor de la Latitud de un Crimen

void crimen::setLocalDescription (string *descripcion_local*)

Fija una Descripcion del lugar a un Crimen.

Parameters:

descripcion_local	es un String, para inicializar el valor de Local_description de un Crimen

void crimen::setLongitude (double *longitud*)

Fija un valor a la variable Longitud de un Crimen.

Parameters:

longitud	es un Double, que indica el valor de la Longitud de un Crimen

void crimen::setPrimaryType (string *primary_type*)

Fija un Tipo Primario a un Crimen.

Parameters:

primary_type	es un String, para inicializar el valor de Primary Type de un Crimen

---

The documentation for this class was generated from the following files:

11 include/**crimen.h**
12 src/**crimen.hxx**

DecrecienteFecha Class Reference

Class recienteFecha operator()

`#include <conjunto.h>`

Public Member Functions

bool **operator()** (const **crimen** &a, const **crimen** &b)

Comparador del funtor,, para ordenar de forma decreciente por fecha.

Detailed Description

Class recienteFecha operator()

Member Function Documentation

bool DecrecienteFecha::operator() (const crimen & *a*, const crimen & *b*)

Comparador del funtor,, para ordenar de forma decreciente por fecha.

Parameters:

a,*tipo*	crimen
b,*tipo*	crimen

The documentation for this class was generated from the following files:

13 include/**conjunto.h**
14 src/**conjunto.hxx**

DecrecienteID Class Reference

Class **DecrecienteID** operator()

`#include <conjunto.h>`

Public Member Functions

bool **operator()** (const **crimen** &a, const **crimen** &b)

Comparador del funtor, para ordenar de forma decreciente por ID.

Detailed Description

Class **DecrecienteID** operator()

Member Function Documentation

bool DecrecienteID::operator() (const crimen & *a*, const crimen & *b*)

Comparador del funtor, para ordenar de forma decreciente por ID.

Parameters:

a,*tipo*	crimen
b,*tipo*	crimen

The documentation for this class was generated from the following files:

15 include/**conjunto.h**
16 src/**conjunto.hxx**

fecha Class Reference

Clase fecha, asociada a la.

`#include <fecha.h>`

Public Member Functions

fecha ()
Constructor por defecto de fecha.

fecha (const string &s)
Constructor con parametro string.

fecha (const **fecha** &s)
Constructor por copia de fecha.

fecha & **operator**= (const **fecha** &f)
Sobrecarga el operador igual (=) con parametro de tipo fecha.

fecha & **operator**= (const string &s)
Sobrecarga el operador igual (=) con parametro de tipo string.

string **toString** () const
Metodo que convierte la clase a string.

bool **operator**== (const **fecha** &f) const
Sobrescribimos operator ==.

bool **operator**< (const **fecha** &f) const
Sobreescribe operador <.

bool **operator**> (const **fecha** &f) const
Sobreescribe operador >

bool **operator**<= (const **fecha** &f) const
Sobreescribe operador <=.

bool **operator**>= (const **fecha** &f) const
Sobreescribe operador >=.

bool **operator!**= (const **fecha** &f) const
Sobreescribe operador !=.

int **getSec** () const
Devuelve el valor de sec.

int **getMin** () const
Devuelve el valor de min.

int **getHour** () const
Devuelve el valor de hou.

int **getDay** () const
Devuelve el valor de day.

int **getMon** () const
Devuelve el valor de mont.

int **getYear** () const

Devuelve el valor de anio.

Friends

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

Sobreescribe operador <<.

Detailed Description

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

Todo:

Implementa esta clase, junto con su documentación asociada

Constructor & Destructor Documentation

fecha::fecha ()

Constructor por defecto de fecha.

fichero de implementacion de la clase fecha

Postcondition:

inicializa todo a 0.

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

Constructor con parametro string.

Parameters:

s	string a convertir , para inicializar la clase

fecha::fecha (const fecha & *s*)

Constructor por copia de fecha.

Parameters:

s	tipo de dato fecha

Member Function Documentation

int fecha::getDay () const[inline]

Devuelve el valor de day.

Returns:
int , dias

Postcondition:
no modifica la clase.

int fecha::getHour () const[inline]

Devuelve el valor de hou.

Returns:
int , hora

Postcondition:
no modifica la clase.

int fecha::getMin () const[inline]

Devuelve el valor de min.

Returns:
int , minutos

Postcondition:
no modifica la clase.

int fecha::getMon () const[inline]

Devuelve el valor de mont.

Returns:
int , mes

Postcondition:
no modifica la clase.

int fecha::getSec () const[inline]

Devuelve el valor de sec.

Returns:

int , segundos

Postcondition:

no modifica la clase.

int fecha::getYear () const[inline]

Devuelve el valor de anio.

Returns:

int , anio

Postcondition:

no modifica la clase.

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

Sobreescribe operador !=.

Parameters:

s	fecha f

Returns:

True si this es distinto de f , false si es distinta

Postcondition:

no modifica la clase.

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

Sobreescribe operador <.

Parameters:

s	fecha f

Returns:

True si this es menor que f , false si es distinta

Postcondition:

no modifica la clase.

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

Sobreescribe operador <=.

Parameters:

s	fecha f

Returns:

True si this es menor o igual que f , false si es distinta

Postcondition:

no modifica la clase.

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

Sobrecarga el operador igual (=) con parametro de tipo fecha.

Parameters:

f	tipo dato fecha

Returns:

Devuelve una refencia con la clase igual a la pasada por parametro

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

Sobrecarga el operador igual (=) con parametro de tipo string.

Parameters:

s	tipo de dato string

Returns:

Devuelve una referencia con la clase creada a partir del string

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

Sobrescribimos operator ==.

Parameters:

fecha	f

Returns:

True si es la misma fecha , false si es distinta

Postcondition:

no modifica la clase.

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

Sobreescribe operador >

Parameters:

s	fecha f

Returns:

True si this es mayor que f , false si es distinta

Postcondition:

no modifica la clase.

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

Sobreescribe operador >=.

Parameters:

s	fecha f

Returns:

True si this es mayor o igual que f , false si es distinta

Postcondition:

no modifica la clase.

string fecha::toString () const

Metodo que convierte la clase a string.

Returns:

string en formato mm/dd/aaaa hh:mm:ss AM/PM

Postcondition:

no modifica la clase.

Friends And Related Function Documentation

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

Sobreescribe operador <<.

Parameters:

ostream	os
fecha	f

Returns:

ostream

Postcondition:

no modifica la clase.

The documentation for this class was generated from the following files:

17 include/**fecha.h**
18 src/**fecha.hxx**

conjunto< CMP >::iterator Class Reference

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

`#include <conjunto.h>`

Public Member Functions

iterator ()
Constructor por defecto de iterator.

iterator (const **iterator** &it)
Constructor de Copia.

iterator (const vector< **conjunto::entrada** >::**iterator** n)
Constructor con parametro iterator tipo vecto<entrada>::iterator.

const **conjunto::entrada** & **operator*** () const
Devuelve el valor del iterador actural.

iterator operator++ (int)
Preincremento ++ ++it.

iterator & **operator**++ ()
Post incremento ++ , it++.

iterator operator-- (int)
Predecremento - -it.

iterator & **operator**-- ()
Post decremento - , it-.

bool **operator**== (const **iterator** &it)
*Sobrecarga operato == de clase **conjunto::iterator**.*

bool **operator!**= (const **iterator** &it)
*Sobrecarga operato != de clase **conjunto::iterator**.*

Friends

class **conjunto**

Detailed Description

template<typename CMP>class conjunto< CMP >::iterator

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

Constructor & Destructor Documentation

template<typename CMP> conjunto< CMP >::iterator::iterator ()

Constructor por defecto de iterator.

template<typename CMP> conjunto< CMP >::iterator::iterator (const iterator & *it*)

Constructor de Copia.

Parameters:

it,tipo	iterator

template<typename CMP> conjunto< CMP >::iterator::iterator (const vector< conjunto::entrada >::iterator *n*)

Constructor con parametro iterator tipo vecto<entrada>::iterator.

Parameters:

n,iterator	tipo vecto<entrada>

Member Function Documentation

template<typename CMP> bool conjunto< CMP >::iterator::operator!= (const iterator & *it*)

Sobrecarga operato != de clase **conjunto::iterator**.

Returns:

false , si son iguales , true , si son distintos

template<typename CMP> const conjunto< CMP >::entrada & conjunto< CMP >::iterator::operator* () const

Devuelve el valor del iterador actural.

Returns:

conjunto::entrada , valor actual del iterator

template<typename CMP> conjunto< CMP >::iterator conjunto< CMP >::iterator::operator+ + (int)

Preincremento ++ ++it.

Returns:

Devuelve el iterador

Postcondition:

Despues de devolverlo , se incremente

template<typename CMP> conjunto< CMP >::iterator & conjunto< CMP >::iterator::operator++ ()

Post incremento ++ , it++.

Returns:

Devuelve el iterador

Postcondition:

Incrementa el iterador para devolverlo

template<typename CMP> conjunto< CMP >::iterator conjunto< CMP >::iterator::operator-- (int)

Predecremento - -it.

Returns:

Devuelve el iterador

Postcondition:

Despues de devolverlo , se decrementa

template<typename CMP> conjunto< CMP >::iterator & conjunto< CMP >::iterator::operator-- ()

Post decremento - , it-.

Returns:

Devuelve el iterador

Postcondition:

Decrementa el iterador para devolverlo

template<typename CMP> bool conjunto< CMP >::iterator::operator== (const iterator & *it*)

Sobrecarga operato == de clase **conjunto::iterator**.

Returns:

true , si son iguales , false , si son distintos

Friends And Related Function Documentation

template<typename CMP> friend class conjunto[friend]

The documentation for this class was generated from the following files:

19 include/**conjunto.h**
20 src/**conjunto.hxx**

File Documentation

include/conjunto.h File Reference

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

Classes

class **CrecienteIUCR**

*Class **CrecienteIUCR** operator()* class **CrecienteID**

*Class **CrecienteID** operator()* class **DecrecienteID**

*Class **DecrecienteID** operator()* class **CrecienteFecha**

*Class **CrecienteFecha** operator()* class **DecrecienteFecha**

Class recienteFecha operator() class **conjunto< CMP >**

Clase conjunto. class **conjunto< CMP >::iterator**

*class iterator forward iterador sobre el conjunto, LECTURA **iterator()** ,**operator*()**, operator++, **operator++(int)** operator=, operator==, operator!=* class **conjunto< CMP >::const_iterator**

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

template<typename CMP > ostream & **operator<<** (ostream &sal, const **conjunto**< CMP > &D)

imprime todas las entradas del conjunto

Function Documentation

template<typename CMP > ostream& operator<< (ostream & *sal*, const conjunto< CMP > & *D*)

imprime todas las entradas del conjunto

Postcondition:

No se modifica el conjunto.

Todo:

implementar esta funcion

include/crimen.h File Reference

```
#include <string>
#include <iostream>
#include <sstream>
#include <stdlib.h>
#include "fecha.h"
#include "../src/crimen.hxx"
```

Classes

class **crimen**

Functions

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

imprime todas las entradas del Crimen

Function Documentation

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

imprime todas las entradas del Crimen

Postcondition:

No se modifica el conjunto.

include/fecha.h File Reference

Fichero con la cabecera de la clase fecha.

```
#include <string>
#include <iostream>
#include <sstream>
#include <stdlib.h>
#include "../src/fecha.hxx"
```

Classes

class **fecha**

Clase fecha, asociada a la. Functions

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

Sobreescribe operador <<.

Detailed Description

Fichero con la cabecera de la clase fecha.

Function Documentation

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

Sobreescribe operador <<.

Parameters:

ostream	os
fecha	f

Returns:

ostream

Postcondition:

no modifica la clase.

src/conjunto.hxx File Reference

`#include "conjunto.h"`

Functions

template<typename CMP > ostream & **operator<<** (ostream &sal, const **conjunto**< CMP > &D)

imprime todas las entradas del conjunto

Function Documentation

template<typename CMP > ostream& operator<< (ostream & *sal*, const conjunto< CMP > & *D*)

imprime todas las entradas del conjunto

Postcondition:

No se modifica el conjunto.

Todo:

implementar esta funcion

src/crimen.hxx File Reference

```
#include "crimen.h"
```

Functions

ostream & **operator**<< (ostream &os, const **crimen** &c)
imprime todas las entradas del Crimen

Function Documentation

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

imprime todas las entradas del Crimen

Postcondition:
No se modifica el conjunto.

src/fecha.hxx File Reference

`#include "fecha.h"`

Functions

std::ostream & **operator**<< (ostream &os, const **fecha** &f)

Sobreescribe operador <<.

Function Documentation

std::ostream& operator<< (ostream & *os*, const fecha & *f*)

Sobreescribe operador <<.

Parameters:

ostream	os
fecha	f

Returns:

ostream

Postcondition:

no modifica la clase.

src/principal.cpp File Reference

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

Functions

template<typename CMP > bool **load** (**conjunto**< CMP > &C, const string &s)

lee un fichero de delitos, linea a linea

int **main** ()

Function Documentation

template<typename CMP > bool load (conjunto< CMP > & *C*, const string & *s*)

lee un fichero de delitos, linea a linea

Parameters:

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

Returns:

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

int main ()

Index

INDEX