CONJUNTO de DELITOS

V0

Generado por Doxygen 1.8.9.1

Viernes, 6 de Noviembre de 2015 13:17:47

Índice

1. Lista de tareas pendientes

Clase conjunto

Implementa esta clase, junto con su documentación asociada

Clase crimen

Implementa esta clase, junto con su documentación asociada

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

implementar esta funcion

2. Índice de clases

2.1. Lista de clases

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

3. Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

4. Documentación de las clases

4.1. Referencia de la Clase conjunto

Clase conjunto.

`#include <conjunto.h>`

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.
- void Ordena ()
- conjunto (const conjunto &d)

  constructor de copia
- pair< conjunto::entrada, bool > 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 identificador. Busca la entrada con id en el conjunto y si la encuentra la borra.
- 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.
- long int ExisteElemento (const long int &ID) const

  Comprueba la existencia de un elemento del conjunto.
- conjunto & operator= (const conjunto &org)

  operador de asignación
- size_type size () const

  numero de entradas en el conjunto
- bool empty () const

  Chequea si el conjunto esta vacio.

Métodos privados

- bool cheq_rep () const

  Chequea el Invariante de la representacion.

Atributos privados

- vector< crimen > vc

Amigas

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

  imprime todas las entradas del conjunto

4.1.1. Descripción detallada

Clase conjunto.

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

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=

Además encontraremos el tipo

```
conjunto::size_type
```

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

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

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;
...
```

Tareas pendientes Implementa esta clase, junto con su documentación asociada

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

4.1.2.1. typedef crimen conjunto::entrada

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

4.1.2.2. typedef unsigned int conjunto::size_type

size_type numero de elementos en el conjunto

4.1.3. Documentación del constructor y destructor

4.1.3.1. conjunto::conjunto ( )

constructor primitivo.

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

constructor de copia

Parámetros

in	*d*	conjunto a copiar

4.1.4. Documentación de las funciones miembro

4.1.4.1. 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
true si el invariante es correcto, falso en caso contrario

4.1.4.2. bool conjunto::empty ( ) const

Chequea si el conjunto esta vacio.

Devuelve

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

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

Borra el delito dado un identificador. 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 éxito. False en caso contrario

Postcondición

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

4.1.4.4. 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 queremos borrar, el resto de los valores no son tenidos en cuenta

Devuelve

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

Postcondición

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

4.1.4.5. long int conjunto::ExisteElemento (const long int & *ID*) const

Comprueba la existencia de un elemento del conjunto.

Parámetros

in	*ID*	cuyo crimen queremos comprobar que existe.

Devuelve

posicion del elemento buscado. -1 si no lo encuentra

4.1.4.6. pair< conjunto::entrada, bool > 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 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

Postcondición

no modifica el conjunto.

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

Parámetros

id	identificador del crimen buscar

Devuelve

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

Postcondición

no modifica el conjunto.

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

4.1.4.7. 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");
```

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

busca los crimenes con el mismo codigo IUCR

Parámetros

icur	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");
```

4.1.4.9. 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 éxito. False en caso contrario

Postcondición

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

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

operador de asignación

Parámetros

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

4.1.4.11. void conjunto::Ordena ( )

4.1.4.12. conjunto::size_type conjunto::size ( ) const

numero de entradas en el conjunto

Postcondición

No se modifica el conjunto.

4.1.5. Documentación de las funciones relacionadas y clases amigas

4.1.5.1. 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

4.1.6. Documentación de los datos miembro

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

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

- conjunto.h
- conjunto.hxx

4.2. Referencia de la Clase crimen

Clase crimen, asociada a la definición de un crimen.

`#include <crimen.h>`

Métodos públicos

- crimen ()
  Constructor primitivo de la clase.
- crimen (const crimen &x)
  Constructor de copia de la clase.
- void setID (long int &id)
  Establecer el ID de un crimen.
- void setCaseNumber (const string &s)
  Establecer el número del caso de un crimen.
- void setDate (const fecha &d)
  Establecer la fecha de un caso.
- void setArrest (bool a)
  Establecer si se produce un arresto o no.

- void setDomestic (bool d)

  Establecer si es un crimen doméstico.
- long int getID () const

  Obtener el ID de un crimen.
- string getCaseNumber () const

  Obtener el número del caso.
- string getDescription () const

  Obtener la descripción del crimen.
- string getIUCR () const

  Obtener el IUCR del crimen.
- void setIUCR (string new_IUCR)

  Asignar un IUCR a un crimen.
- void setDescription (string new_Descr)

  Asignar una descripción a un crimen.
- fecha getDate () const

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

  Copia en un crimen los datos de otro pasado como string.
- crimen & operator= (const crimen &c)

  Iguala dos crímenes.
- bool operator== (const crimen &x) const

  Compara si son iguales dos crímenes.
- bool operator< (const crimen &x) const

  Compara si un conjunto es mayor que otro.

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 &)

  Imprime todas las entradas del crimen.

4.2.1. Descripción detallada

Clase crimen, asociada a la definición de un crimen.

crimen::crimen, ... Descripción contiene toda la información asociada a un crimen.

Tareas pendientes Implementa esta clase, junto con su documentación asociada

4.2.2. Documentación del constructor y destructor

4.2.2.1. crimen::crimen ( )

Constructor primitivo de la clase.

Clase crimen, asociada a la definición de un crimen.

crimen::crimen, ... Descripción contiene toda la información asociada a un crimen.

/**Constructor primitivo de la clase.

Postcondición

Se crea un nuevo objeto del tipo crimen.

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

Constructor de copia de la clase.

Parámetros

in	*c*	crimen a copiar.
in	*x*	Crimen del que se copian los datos.

Postcondición

Se crea un nuevo objeto del tipo crimen con los datos del objeto x.

4.2.3. Documentación de las funciones miembro

4.2.3.1. string crimen::getCaseNumber ( ) const

Obtener el número del caso.

Devuelve

Devuelve el número del caso.
El número del caso.

4.2.3.2. fecha crimen::getDate ( ) const

Devuelve la fecha del crimen.

Devuelve

Devuelve la fecha del crimen.
La fecha del crimen.

4.2.3.3. string crimen::getDescription ( ) const

Obtener la descripción del crimen.

Devuelve

string con la descripción del caso.

4.2.3.4. long crimen::getID ( ) const

Obtener el ID de un crimen.

Devuelve

Devuelve el ID.
El ID del crimen.

4.2.3.5. string crimen::getIUCR ( ) const

Obtener el IUCR del crimen.

Devuelve

string con el IUCR del caso.

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

Compara si un conjunto es mayor que otro.

Compara si un conjunto es menor que otro.

Parámetros

x	Crimen a comparar.

Devuelve

Devuelve true si x es mayor que el que lo llama. False en otro caso.

Parámetros

in	*x*	Crimen a comparar.

Devuelve

true si x es mayor que el que lo llama. False en otro caso.

4.2.3.7. crimen & crimen::operator= (const string & *datos*)

Copia en un crimen los datos de otro pasado como string.

Parámetros

in	*string*	con los datos a copiar de otro crimen.

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

Iguala dos crímenes.

Parámetros

c	Crimen a copiar en el que lo llama.

Devuelve

Devuelve una copia del crimen.

Parámetros

in	*c*	Crimen a copiar en el que lo llama.

Devuelve

Una copia del crimen.

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

Compara si son iguales dos crímenes.

Parámetros

x	Crimen a comparar.

Devuelve

Devuelve true si son iguales y false si no lo son.

Parámetros

in	*x*	Crimen a comparar.

Devuelve

true si son iguales y false si no lo son.

4.2.3.10. void crimen::setArrest (bool *a*)

Establecer si se produce un arresto o no.

Parámetros

	a	Se produce arresto -> True / No se produce arresto -> False
in	*a*	Se produce arresto -> True / No se produce arresto -> False

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

Establecer el número del caso de un crimen.

Parámetros

	in	s Número del caso.
in	*s*	Número del caso.

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

Establecer la fecha de un caso.

Parámetros

	d	Fecha a establecer.
in	*d*	Fecha a establecer.

4.2.3.13. void crimen::setDescription (string *new_Descr*)

Asignar una descripción a un crimen.

Parámetros

in	*string*	con la descripción a asignar.

4.2.3.14. void crimen::setDomestic (bool *d*)

Establecer si es un crimen doméstico.

Parámetros

	d	Es crimen doméstico -> True / No es crimen doméstico -> False
in	*d*	Es crimen doméstico -> True / No es crimen doméstico -> False

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

Establecer el ID de un crimen.

Parámetros

in	*id*	ID a establecer.

4.2.3.16. void crimen::setIUCR (string *new_IUCR*)

Asignar un IUCR a un crimen.

Parámetros

in	*string*	con el IUCR a asignar.

4.2.4. Documentación de las funciones relacionadas y clases amigas

4.2.4.1. ostream& operator<< (ostream & , const crimen & *x*) [friend]

Imprime todas las entradas del crimen.

Postcondición

No se modifica el crimen original.

4.2.5. Documentación de los datos miembro

4.2.5.1. bool crimen::Arrest [private]

4.2.5.2. string crimen::CaseNumber [private]

4.2.5.3. fecha crimen::Date [private]

4.2.5.4. string crimen::Description [private]

4.2.5.5. bool crimen::Domestic [private]

4.2.5.6. long int crimen::ID [private]

4.2.5.7. string crimen::IUCR [private]

4.2.5.8. double crimen::Latitude [private]

4.2.5.9. string crimen::LocationDescription [private]

4.2.5.10. double crimen::Longitude [private]

4.2.5.11. string crimen::PrimaryType [private]

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

- crimen.h
- crimen.hxx

4.3. Referencia de la Clase fecha

`#include <fecha.h>`

Métodos públicos

- fecha ()

  Constructor primitivo de la clase.
- fecha (const string &s)

  Constructor de copia de la clase.
- fecha & operator= (const fecha &f)

  operador de asignación
- fecha & operator= (const string &s)

  operador de asignación
- string toString () const

  Muestra el valor de sus atributos.
- string DarFormato (int param) const

  Imprime datos en formato correcto (horas, minutos, etc)
- bool operator== (const fecha &f) const

  Compara si son iguales dos fechas.
- bool operator< (const fecha &f) const

  Compara si una fecha es mayor que otra.
- bool operator> (const fecha &f) const

  Compara si una fecha es menor que otra.
- bool operator<= (const fecha &f) const

  Compara si una fecha es mayor o igual que otra.
- bool operator>= (const fecha &f) const

  Compara si una fecha es menor o igual que otra.
- bool operator!= (const fecha &f) const

  Es un comparador de desigualdad.

Atributos privados

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

Amigas

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

  Imprime todas las entradas de las fechas.

4.3.1. Documentación del constructor y destructor

4.3.1.1. fecha::fecha ( )

Constructor primitivo de la clase.

fichero de implementacion de la clase fecha

Constructor sin parametros de la clase.

Postcondición

Se crea un nuevo objeto fecha con parametros por defecto.

4.3.1.2. fecha::fecha (const string & *x*)

Constructor de copia de la clase.

Parámetros

in	*s*	string a copiar:
in	*x*	fecha del que se copian los datos.

Postcondición

Se crea un nuevo objeto del tipo fehca con los datos del objeteo x.

4.3.2. Documentación de las funciones miembro

4.3.2.1. string fecha::DarFormato (int *param*) const

Imprime datos en formato correcto (horas, minutos, etc)

Parámetros

in	*param*	entero que se comprueba y/o formatea

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

Es un comparador de desigualdad.

Compara si una fecha.

Parámetros

in	*f*	Fecha a comparar.

Devuelve

Devuelve true cuando f es distinta del que la llama.False en otro caso.

Parámetros

in	*f*	fecha a comparar.

Devuelve

true si es mayor.

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

Compara si una fecha es mayor que otra.

Compara si una fecha es menor que otra.

Parámetros

in	*f*	Fecha a comparar.

Devuelve

Devuelve true si f es mayor que el que lo llama. False en otro caso.

Parámetros

in	*f*	fecha a comparar.

Devuelve

true si es menor.

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

Compara si una fecha es mayor o igual que otra.

Compara si una fecha es menor o igual que otra.

Parámetros

in	*f*	Fecha a comparar.

Devuelve

Devuelve true si f es mayor o igual que el que lo llama.False en otro caso.

Parámetros

in	*f*	fecha a comprar.

Devuelve

true si es menor.

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

operador de asignación

Iguala dos fechas.

Parámetros

in	*f*	fecha a copiar.

Postcondición

Crea una fecha duplicada exacta de f

Parámetros

in	*f*	fecha a copiar en el que lo llama.

Devuelve

Una copia de la fecha.

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

operador de asignación

Iguala dos fechas.

Parámetros

in	*s*	string a copiar.
in	*s*	fecha a copiar en el que lo llama.

Devuelve

Una copia de la fecha.

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

Compara si son iguales dos fechas.

Parámetros

in	*f*	Fecha a comparar.

Devuelve

Devuelve true si son iguales y false si no lo son.

Parámetros

in	*f*	fecha a comparar.

Devuelve

true si son iguales y false si no lo son.

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

Compara si una fecha es menor que otra.

Compara si una fecha es mayor que otra.

Parámetros

in	*f*	Fecha a comparar.

Devuelve

Devuelve true si f es menor que el que lo llama. False en otro caso.

Parámetros

in	*f*	fecha a comprar.

Devuelve

true si es mayor.

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

Compara si una fecha es menor o igual que otra.

Compara si una fecha es mayor que otra.

Parámetros

in	*f*	Fecha a comparar.

Devuelve

Devuelve true si f es menor o igual que el que lo llama.False en otro caso.

Parámetros

in	*f*	fecha a comprar.

Devuelve

true si es mayor.

4.3.2.10. string fecha::toString ( ) const

Muestra el valor de sus atributos.

Copiar los datos de una fecha a un string.

Devuelve

Devuelve la fecha convertida a un string

Postcondición

Se crea un objeto string con los datos de fecha

4.3.3. Documentación de las funciones relacionadas y clases amigas

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

Imprime todas las entradas de las fechas.

Postcondición

No se modifica la fecha original

Parámetros

in	*os*	flujo
in	*f*	fecha a mostrar.

Devuelve

La fecha por salida estandar.

4.3.4. Documentación de los datos miembro

4.3.4.1. int fecha::hour [private]

4.3.4.2. int fecha::mday [private]

4.3.4.3. int fecha::min [private]

4.3.4.4. int fecha::mon [private]

4.3.4.5. int fecha::sec [private]

4.3.4.6. int fecha::year [private]

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

- fecha.h
- fecha.hxx

5. Documentación de archivos

5.1. Referencia del Archivo conjunto.h

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

Clases

- class conjunto

  Clase conjunto.

Funciones

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

  imprime todas las entradas del conjunto

5.1.1. Documentación de las funciones

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

imprime todas las entradas del conjunto

Postcondición

No se modifica el conjunto.

Tareas pendientes implementar esta funcion

5.2. Referencia del Archivo conjunto.hxx

```
#include "conjunto.h"
#include "crimen.h"
#include "fecha.h"
```

5.3. Referencia del Archivo crimen.h

```
#include <string>
#include <vector>
#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 &)

5.3.1. Documentación de las funciones

5.3.1.1. ostream& operator<< (ostream & , const crimen & *x*)

Postcondición

No se modifica el crimen original.

5.4. Referencia del Archivo crimen.hxx

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

Funciones

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

5.4.1. Documentación de las funciones

5.4.1.1. ostream& operator<< (ostream & , const crimen & *x*)

Postcondición

No se modifica el crimen original.

5.5. Referencia del Archivo fecha.h

```
#include <vector>
#include <string>
#include <iostream>
#include "fecha.hxx"
```

Clases

- class fecha

Funciones

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

5.5.1. Documentación de las funciones

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

imprime fecha con el formato mm/dd/aaaa hh:mm:ss AM/PM

Imprime todas las entradas de las fechas.

Postcondición

No se modifica la fecha original

imprime fecha con el formato mm/dd/aaaa hh:mm:ss AM/PM

Imprime todas las entradas de las fechas.

Parámetros

in	*os*	flujo
in	*f*	fecha a mostrar.

Devuelve

La fecha por salida estandar.

5.6. Referencia del Archivo fecha.hxx

Funciones

- ostream & operator<< (ostream &os, const fecha &f)
  Muestra por pantalla un objeto del tipo fecha.

5.6.1. Documentación de las funciones

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

Muestra por pantalla un objeto del tipo fecha.

imprime fecha con el formato mm/dd/aaaa hh:mm:ss AM/PM

Imprime todas las entradas de las fechas.

Parámetros

in	*os*	flujo
in	*f*	fecha a mostrar.

Devuelve

La fecha por salida estandar.

5.7. Referencia del Archivo principal.cpp

```
#include "conjunto.h"
```

Funciones

- bool load (conjunto &C, const string &S)

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

5.7.1. Documentación de las funciones

5.7.1.1. 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

5.7.1.2. int main ( )

Índice alfabético