CONJUNTO de DELITOS

V0

Generado por Doxygen 1.8.6

Martes, 3 de Noviembre de 2015 14:59:55

Índice

1 Documentación Práctica **1**

1.1 Introducción . . . 1

1.1.1 Conjunto de Datos . . . 2

1.2 "Fecha Límite de Entrega" . . . 2

1.3 Crimen . . . 3

1.4 Conjunto como TDA contenedor de información . . . 3

1.5 "Se Entrega / Se Pide" . . . 3

1.5.1 Se entrega . . . 3

1.5.2 Se Pide . . . 4

1.5.3 Eficiencia . . . 4

1.6 Representación . . . 6

1.6.1 Función de Abstracción : . . . 6

1.6.2 Invariante de la Representación: . . . 6

1.7 "Fecha Límite de Entrega" . . . 7

2 Índice de clases **7**

2.1 Lista de clases . . . 7

3 Indice de archivos **7**

3.1 Lista de archivos . . . 7

4 Documentación de las clases **7**

4.1 Referencia de la Clase conjunto . . . 7

4.1.1 Descripción detallada . . . 9

4.1.2 Documentación de los 'Typedef' miembros de la clase . . . 9

4.1.3 Documentación del constructor y destructor . . . 9

4.1.4 Documentación de las funciones miembro . . . 10

4.1.5 Documentación de las funciones relacionadas y clases amigas . . . 12

4.1.6 Documentación de los datos miembro . . . 12

4.2 Referencia de la Clase crimen . . . 13

4.2.1 Descripción detallada . . . 14

4.2.2 Documentación del constructor y destructor . . . 15

4.2.3 Documentación de las funciones miembro . . . 15

4.2.4 Documentación de las funciones relacionadas y clases amigas . . . 18

4.2.5 Documentación de los datos miembro . . . 20

4.3 Referencia de la Clase fecha . . . 20

4.3.1 Descripción detallada . . . 21

4.3.2 Documentación del constructor y destructor . . . 21

4.3.3 Documentación de las funciones miembro . . . 22

4.3.4 Documentación de las funciones relacionadas y clases amigas . . . 24
4.3.5 Documentación de los datos miembro . . . 24

5 Documentación de archivos 24

5.1 Referencia del Archivo conjunto.h . . . 24
5.1.1 Descripción detallada . . . 25
5.1.2 Documentación de las funciones . . . 25
5.2 Referencia del Archivo conjunto.hxx . . . 25
5.2.1 Descripción detallada . . . 25
5.2.2 Documentación de las funciones . . . 25
5.3 Referencia del Archivo crimen.h . . . 26
5.3.1 Descripción detallada . . . 26
5.3.2 Documentación de las funciones . . . 26
5.4 Referencia del Archivo crimen.hxx . . . 26
5.4.1 Descripción detallada . . . 27
5.4.2 Documentación de las funciones . . . 27
5.5 Referencia del Archivo documentacion.dox . . . 27
5.6 Referencia del Archivo fecha.h . . . 27
5.6.1 Descripción detallada . . . 27
5.6.2 Documentación de las funciones . . . 28
5.7 Referencia del Archivo fecha.hxx . . . 29
5.7.1 Descripción detallada . . . 29
5.7.2 Documentación de las funciones . . . 29
5.8 Referencia del Archivo principal.cpp . . . 30
5.8.1 Documentación de las funciones . . . 30

Índice 31

1. Documentación Práctica

Versión

v1

Autor

Alejandro Campoy Nieves
David Criado Ramón

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 59
  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 fecha & f); //Constructor de copia
 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.

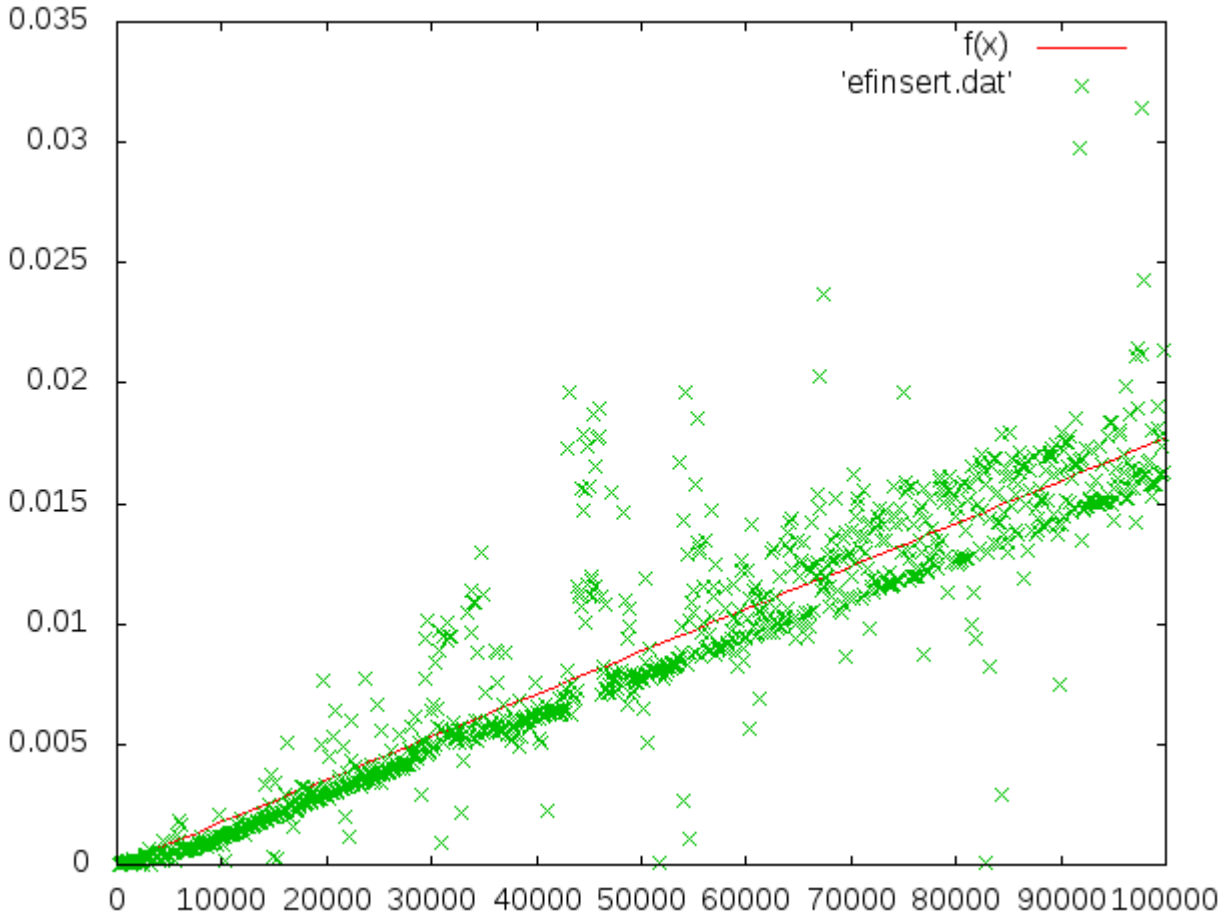
- conjunto.h Especificación del TDA conjunto.
- conjunto.hxx plantilla de fichero donde debemos implementar el conjunto.
- crimen.h Plantilla para la especificación del TDA crimen
- crimen.hxx plantilla de fichero donde debemos implementar el crimen
- fecha.h Plantilla para la especificación del TDA 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

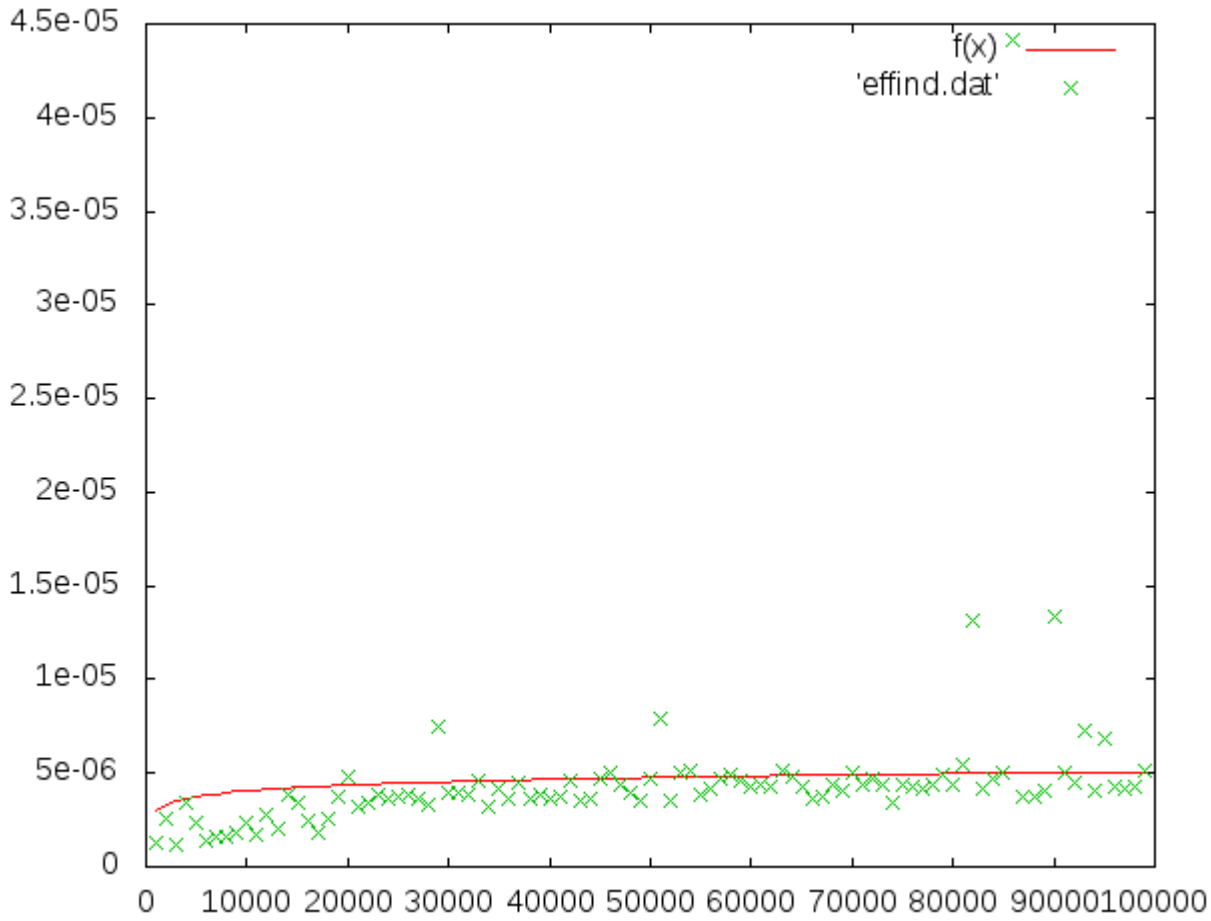
- Diseñar la función de abstracción e invariante de la representación del tipo fecha (Mirar clase fecha).
- Diseñar la función de abstracción e invariante de la representación del tipo crimen (Mirar clase crimen).
- Se pide implementar el código asociado a los ficheros .hxx. (Comprobar Ficheros).
- Analizar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

1.5.3. Eficiencia

- Análisis de la inserción, en el peor de los casos implicará un movimiento O(n) ya que implicaría desplazar la memoria de el conjunto entero al insertar al inicio. Un gráfico de la eficiencia empírica de la híbrida sería:



- Análisis de la búsqueda, implicará un movimiento O(log n) ya que para encontrarlos se utiliza el conocido método de la búsqueda binaria sería:



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 < C.vc.size()-1 se cumple
    C.vc[i].ID<= C.vc[i+1].ID
```

1.7. "Fecha Límite de Entrega"

La fecha límite de entrega será el 6 de Noviembre.

2. Índice de clases

2.1. Lista de clases

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

conjunto
Clase conjunto 7

crimen
Clase crimen, contiene toda la información asociada a un crimen 13

fecha
Clase fecha, contiene toda la información asociada a una fecha con el formato mm/dd/aaaa hh:mm:ss AM/PM 20

3. Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

conjunto.h
Declaración de la clase conjunto 24

conjunto.hxx
Implementación de la clase conjunto 25

crimen.h
Declaración de la clase crimen 26

crimen.hxx
Implementación de la clase crimen 26

fecha.h
Declaración de la clase fecha 27

fecha.hxx
Implementación de la clase fecha 29

principal.cpp 30

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 por defecto de la clase conjunto (Conjunto vacio).
- conjunto (const conjunto &d)

  Constructor de copia. Conjunto a copiar.
- pair< conjunto::entrada, bool > find (const long int &id) const

  Busca una entrada dada su id en el conjunto.
- conjunto findIUCR (const string &iucr) const

  Devuelve un conjunto con todos los crímenes con el mismo IUCR.
- conjunto findDESCR (const string &descr) const

  Devuelve un conjunto con todos los crímenes con la misma descripción.
- bool insert (const conjunto::entrada &e)

  Inserta una entrada en la posición apropiada del conjunto.
- bool erase (const long int &id)

  Elimina un elemento del conjunto dado su identificador.
- bool erase (const conjunto::entrada &e)

  Elimina un elemento del conjunto dado un crimen.
- conjunto & operator= (const conjunto &org)

  Sobrecarga del operador de asignación.
- size_type size () const

  Devuelve el tamaño del conjunto.
- bool empty () const

  Comprueba si el conjunto está vacío.

Métodos privados

- bool cheq_rep () const

  Comprueba el invariante de la representación de la clase conjunto.

Atributos privados

- vector< crimen > vc

Amigas

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

  Sobrecarga del operador de extracción de flujos para la clase 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;
...
```

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 por defecto de la clase conjunto (Conjunto vacio).

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

Constructor de copia. Conjunto a copiar.

4.1.4. Documentación de las funciones miembro

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

Comprueba el invariante de la representación de la clase conjunto.

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 se cumple el invariante, false en caso contrario.
True si se cumple el invariante, false en caso contrario.

4.1.4.2. bool conjunto::empty ( ) const

Comprueba si el conjunto está vacío.

Devuelve

True si el conjunto está vacío, false en caso contrario.

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

Elimina un elemento del conjunto dado su identificador.

Parámetros

id	ID de la entrada a eliminar del conjunto.

Devuelve

True si se ha podido eliminar, false en caso contrario (no se encontró un crimen con esa id).

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

Elimina un elemento del conjunto dado un crimen.

Parámetros

e	Crimen a eliminar en el conjunto.

Devuelve

True si se ha podido eliminar, false en caso contrario (no se encontró el crimen).

4.1.4.5. pair< conjunto::entrada, bool > conjunto::find (const long int & *id*) const

Busca una entrada dada su id en el conjunto.

Parámetros

id	ID del crimen a buscar.

Devuelve

Par (crimen,verdadero) si se encuentra o (–,falso) si no se encuentra.

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

Parámetros

id	ID del crimen a buscar.

Devuelve

Par (crimen,verdadero) si se encuentra o (–,falso) si no se encuentra.

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

Devuelve un conjunto con todos los crímenes con la misma descripción.

Parámetros

descr	Descripción a buscar

Devuelve

Conjunto con todos los crímenes con la descripción proporcionada.

```
Uso
   conjunto C, A;
   ....
   A = C.findDESCR("BATTERY");
```

Parámetros

descr	Descripción a buscar.

Devuelve

Conjunto con todos los crímenes con la descripción proporcionada.

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

Devuelve un conjunto con todos los crímenes con el mismo IUCR.

Parámetros

iucr	IUCR a buscar.

Devuelve

Conjunto con crímenes con el IUCR proporcionado.

```
Uso
   conjunto C, A;
   ....
   A = C.findIUCR("0460");
```

Parámetros

iucr	IUCR a buscar.

Devuelve

Conjunto con crímenes con el IUCR proporcionado.

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

Inserta una entrada en la posición apropiada del conjunto.

Parámetros

e	Crimen a introducir.

Devuelve

Si se ha insertado correctamente o no (ya existía un crimen con dicha ID).

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

Sobrecarga del operador de asignación.

Parámetros

org	Conjunto a asignar.

Devuelve

Referencia al objeto de la clase (this).

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

Devuelve el tamaño del conjunto.

Devuelve

Tamaño del conjunto

4.1.5. Documentación de las funciones relacionadas y clases amigas

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

Sobrecarga del operador de extracción de flujos para la clase conjunto.

Parámetros

sal	Flujo de salida.
D	Conjunto a mostrar.

Devuelve

Referencia al flujo de salida.

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, contiene toda la información asociada a un crimen.

`#include <crimen.h>`

Métodos públicos

- crimen ()

  Constructor por defecto. Id, longitud y latitud a 0, fecha nula, strings vacios.
- crimen (const crimen &x)

  Constructor de copia.
- crimen & operator= (const string &s)

  Sobrecarga del operador de asignación a partir de string.
- crimen & operator= (const crimen &c)

  Sobrecarga del operador de asignación.
- void setID (const long int &id)

  Modificador del atributo identificador.
- void setCaseNumber (const string &s)

  Modificador del atributo número de caso.
- void setDate (const fecha &d)

  Modificador del atributo fecha.
- void setIUCR (const string &iucr)

  Modificador del atributo IUCR.
- void setArrest (const bool &a)

  Modificador del atributo arresto.
- void setDomestic (const bool &d)

  Modificador del atributo doméstico.
- void setDESCR (const string &s)

  Modificador del atributo descripción.
- void setPrimaryType (const string &s)

  Modificador del atributo tipo principal.
- void setLocation (const string &s)

  Modificador del tipo de ubicación.
- void setLatitude (const double &d)

  Modificador del atributo latitud.
- void setLongitude (const double &d)

  Modificador del atributo longitud.
- long int getID () const

  Acceso al atributo ID.
- string getCaseNumber () const

  Accede al número de caso.
- fecha getDate () const

  Accede a la fecha.
- string getDESCR () const

  Accede a la descripción.
- string getIUCR () const

  Accede al IUCR.

- string getLocation () const

  Accede al tipo de ubicación.
- string getPrimaryType () const

  Accede al tipo primario.
- bool getArrest () const

  Accede al atributo arresto.
- bool getDomestic () const

  Accede al atributo doméstico.
- double getLatitude () const

  Accede al atributo latitud.
- double getLongitude () const

  Accede al atributo longitud.
- bool operator== (const crimen &x) const

  Sobrecarga del operador de igualdad.
- bool operator< (const crimen &x) const

  Sobrecarga del operador menor.

Atributos privados

- long int ID
- string case_number
- fecha date
- string IUCR
- string primary_type
- string description
- string location
- bool arrest
- bool domestic
- double latitude
- double longitude

Amigas

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

  Sobrecarga del operador de extracción de flujo.

4.2.1. Descripción detallada

Clase crimen, contiene toda la información asociada a un crimen.

Invariante

ID $>$ 0 $

Además su función de abstracción es:

AF: Representacion $\rightarrow$ Abstraccion

Dada la tupla $(ID, CaseNumber, Date, IUCR, PrimaryType, Description, Location, Arrest, Domestic, Latitude, Longitude)$ $\Rightarrow$ f

C es un objeto abstracto al que denominamos conjunto y que instanciamos como un entero (ID), 1 fecha (fecha), 2 double (Latitud y Longitud), 5 string (Número de caso, IUCR, Tipo Primario, Descripción y Localización) y 2 valores booleanos (Arresto y Doméstico).

4.2.2. Documentación del constructor y destructor

4.2.2.1. crimen::crimen ( )

Constructor por defecto. Id, longitud y latitud a 0, fecha nula, strings vacios.

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

Constructor de copia.

Parámetros

x	Objeto de tipo crimen a copiar.

4.2.3. Documentación de las funciones miembro

4.2.3.1. bool crimen::getArrest ( ) const

Accede al atributo arresto.

Devuelve

Atributo Arrest.

4.2.3.2. string crimen::getCaseNumber ( ) const

Accede al número de caso.

Devuelve

Atributo CaseNumber

4.2.3.3. fecha crimen::getDate ( ) const

Accede a la fecha.

Devuelve

Atributo Fecha

4.2.3.4. string crimen::getDESCR ( ) const

Accede a la descripción.

Devuelve

Atributo description

4.2.3.5. bool crimen::getDomestic ( ) const

Accede al atributo doméstico.

Devuelve

Atributo Domestic.

4.2.3.6. long int crimen::getID ( ) const

Acceso al atributo ID.

Devuelve

Atributo ID

4.2.3.7. string crimen::getIUCR ( ) const

Accede al IUCR.

Devuelve

Atributo IUCR

4.2.3.8. double crimen::getLatitude ( ) const

Accede al atributo latitud.

Devuelve

Atributo Latitude.

4.2.3.9. string crimen::getLocation ( ) const

Accede al tipo de ubicación.

Devuelve

Atributo Location Description.

4.2.3.10. double crimen::getLongitude ( ) const

Accede al atributo longitud.

Devuelve

Atributo Longitude.

4.2.3.11. string crimen::getPrimaryType ( ) const

Accede al tipo primario.

Devuelve

Atributo Primary Type

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

Sobrecarga del operador menor.

Parámetros

x	Crimen a comparar.

Devuelve

Devuelve si el crimen es menor.

4.2.3.13. crimen & crimen::operator= (const string & *s*)

Sobrecarga del operador de asignación a partir de string.

Constructor a partir de string.

Parámetros

s	Cadena a leer.

4.2.3.14. crimen & crimen::operator= (const crimen & *x*)

Sobrecarga del operador de asignación.

Parámetros

x	crimen a asignar.

Devuelve

devuelve referencia al objeto de la clase (this).

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

Sobrecarga del operador de igualdad.

Parámetros

x	crimen a comparar.

Devuelve

True si son iguales, false en caso contrario.

4.2.3.16. void crimen::setArrest (const bool & *a*)

Modificador del atributo arresto.

Parámetros

a	Valor a sobreescribir en el atributo arrest.

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

Modificador del atributo número de caso.

Parámetros

s	Valor a sobreescribir en el atributo Case Number.

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

Modificador del atributo fecha.

Parámetros

d	Valor a sobreescribir en el atributo Date.

4.2.3.19. void crimen::setDESCR (const string & *s*)

Modificador del atributo descripción.

Parámetros

s	Valor a sobreescribir en el atributo description.

4.2.3.20. void crimen::setDomestic (const bool & *d*)

Modificador del atributo doméstico.

Parámetros

d	Valor a sobreescribir en el atributo domestic.

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

Modificador del atributo identificador.

Parámetros

id	Valor a sobreescribir en el atributo ID.

4.2.3.22. void crimen::setIUCR (const string & *iucr*)

Modificador del atributo IUCR.

Parámetros

iucr	Valor a sobreescribir en el atributo iucr.

4.2.3.23. void crimen::setLatitude (const double & *d*)

Modificador del atributo latitud.

Parámetros

d	Valor a sobreescribir en el atributo latitude.

4.2.3.24. void crimen::setLocation (const string & *s*)

Modificador del tipo de ubicación.

Parámetros

s	Valor a sobreescribir en el atributo Location Description.

4.2.3.25. void crimen::setLongitude (const double & *d*)

Modificador del atributo longitud.

Parámetros

d	valor a sobreescribir en el atributo longitude.

4.2.3.26. void crimen::setPrimaryType (const string & *s*)

Modificador del atributo tipo principal.

Parámetros

s	valor a sobreescribir en el atributo Primary Type.

4.2.4. Documentación de las funciones relacionadas y clases amigas

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

Sobrecarga del operador de extracción de flujo.

Parámetros

os	Flujo de salida.
x	Crimen a mostrar.

Devuelve

Referencia al flujo de salida.

4.2.5. Documentación de los datos miembro

4.2.5.1. bool crimen::arrest [private]

4.2.5.2. string crimen::case_number [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::location [private]

4.2.5.10. double crimen::longitude [private]

4.2.5.11. string crimen::primary_type [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

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

`#include <fecha.h>`

Métodos públicos

- fecha ()

  Constructor por defecto de la clase fecha. Inicializa los datos a "00-00-0000 00:00:00 AM".
- fecha (const fecha &F)

  Constructor de copia.
- fecha (const string &s)

  Crea un objeto fecha a partir de un string.
- fecha & operator= (const fecha &f)

  Operador de asignación para fecha.
- fecha & operator= (const string &s)

  Operador de asignación para la clase fecha con fechas en formato string.
- string toString () const

Da la verisón string de la clase fecha.

- bool operator== (const fecha &f) const

  Sobrecarga del operador de igualdad para la clase fecha.
- bool operator< (const fecha &f) const

  Sobrecarga del operador de menor para la clase fecha.
- bool operator> (const fecha &f) const

  Sobrecarga del operador de mayor para la clase fecha.
- bool operator<= (const fecha &f) const

  Sobrecarga del operador de menor o igual para la clase fecha.
- bool operator>= (const fecha &f) const

  Sobrecarga del operador de mayor o igual para la clase fecha.
- bool operator!= (const fecha &f) const

  Sobrecarga del operador de igualdad para la clase fecha.

Atributos privados

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

Amigas

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

  Sobrecarga del operador << para los flujos de salida de la clase.

4.3.1. Descripción detallada

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

- Toda fecha debe cumplir que:

  Invariante

  $0 \leq$ hour(hora) < 24
  $0 \leq$ min(minuto) < 60
  $0 \leq$ sec(segundo) < 60
  $0 \leq$ mday(día) ≤ 31
  $0 \leq$ mon(mes) ≤ 12

  Además su función de abstracción es:

  AF: Representacion $\rightarrow$ Abstraccion

  Dada la tupla $(anio, mes, dia, hora, minuto, segundo, am/pm) \Rightarrow$ f

  F es un objeto abstracto que se instacia como una tupla de año, mes, día, hora, minutos y segundos.

4.3.2. Documentación del constructor y destructor

4.3.2.1. fecha::fecha ( )

Constructor por defecto de la clase fecha. Inicializa los datos a "00-00-0000 00:00:00 AM".

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

Constructor de copia.

Parámetros

x	Objeto de tipo fecha a copiar.

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

Crea un objeto fecha a partir de un string.

Parámetros

s	es un string con el formato "MM-DD-AAAA HH:MM:SS AM".
s	es un string con el formato "MM-DD-AAAA HH:MM:SS AM"

4.3.3. Documentación de las funciones miembro

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

Sobrecarga del operador de igualdad para la clase fecha.

Parámetros

f	como fecha a comparar

Devuelve

True si no son iguales y false si son iguales

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

Sobrecarga del operador de menor para la clase fecha.

Parámetros

f	como fecha a comparar

Devuelve

True si la clase es menor que f y false en caso contrario

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

Sobrecarga del operador de menor o igual para la clase fecha.

Parámetros

f	como fecha a comparar

Devuelve

True si la clase es menor o igual que f y false en caso contrario

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

Operador de asignación para fecha.

Parámetros

f	objeto de tipo fecha.

Devuelve

Referencia al objeto fecha (this).

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

Operador de asignación para la clase fecha con fechas en formato string.

Parámetros

s	como string fecha en formato "MM-DD-AAAA HH:MM:SS AM".

Devuelve

Referencia al objeto fecha (this).

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

Sobrecarga del operador de igualdad para la clase fecha.

Parámetros

f	como fecha a comparar

Devuelve

True si son iguales y false en caso contrario

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

Sobrecarga del operador de mayor para la clase fecha.

Parámetros

f	como fecha a comparar

Devuelve

True si la clase es mayor que f y false en caso contrario

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

Sobrecarga del operador de mayor o igual para la clase fecha.

Parámetros

f	como fecha a comparar

Devuelve

True si la clase es mayor o igual que f y false en caso contrario

4.3.3.9. string fecha::toString ( ) const

Da la verisón string de la clase fecha.

Devuelve

Devuelve el string de la fecha en formato "MM-DD-AAAA HH:MM:SS AM"

4.3.4. Documentación de las funciones relacionadas y clases amigas

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

Sobrecarga del operador << para los flujos de salida de la clase.

Parámetros

os	como flujo de entrada dado por referencia para indicar las operaciones a realizar.
f	como fecha dado por referencia a partir de la cual se mete sus datos en el flujo.

Devuelve

Nos devuelve dicho flujo pasado como primer parámetro.

4.3.5. Documentación de los datos miembro

4.3.5.1. int fecha::hour [private]

4.3.5.2. int fecha::mday [private]

4.3.5.3. int fecha::min [private]

4.3.5.4. int fecha::mon [private]

4.3.5.5. int fecha::sec [private]

4.3.5.6. 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

Declaración de la clase conjunto.

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

Clases

- class conjunto

  Clase conjunto.

Funciones

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

Sobrecarga del operador de extracción de flujos para la clase conjunto.

5.1.1. Descripción detallada

Declaración de la clase conjunto.

Autor

Alejandro Campoy Nieves
David Criado Ramón

5.1.2. Documentación de las funciones

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

Sobrecarga del operador de extracción de flujos para la clase conjunto.

Parámetros

sal	Flujo de salida.
D	Conjunto a mostrar.

Devuelve

Referencia al flujo de salida.

5.2. Referencia del Archivo conjunto.hxx

Implementación de la clase conjunto.

Funciones

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

  Sobrecarga del operador de extracción de flujos para la clase conjunto.

5.2.1. Descripción detallada

Implementación de la clase conjunto.

Autor

Alejandro Campoy Nieves
David Criado Ramón

5.2.2. Documentación de las funciones

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

Sobrecarga del operador de extracción de flujos para la clase conjunto.

Parámetros

sal	Flujo de salida.
D	Conjunto a mostrar.

Devuelve

Referencia al flujo de salida.

5.3. Referencia del Archivo crimen.h

Declaración de la clase crimen.

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

Clases

- class crimen

  Clase crimen, contiene toda la información asociada a un crimen.

Funciones

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

  Sobrecarga del operador de extracción de flujo.

5.3.1. Descripción detallada

Declaración de la clase crimen.

Autor

Alejandro Campoy Nieves
David Criado Ramón

5.3.2. Documentación de las funciones

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

Sobrecarga del operador de extracción de flujo.

Parámetros

os	Flujo de salida.
x	Crimen a mostrar.

Devuelve

Referencia al flujo de salida.

5.4. Referencia del Archivo crimen.hxx

Implementación de la clase crimen.

Funciones

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

  Sobrecarga del operador de extracción de flujo.

5.4.1. Descripción detallada

Implementación de la clase crimen.

Autor

Alejandro Campoy Nieves
David Criado Ramón

5.4.2. Documentación de las funciones

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

Sobrecarga del operador de extracción de flujo.

Parámetros

os	Flujo de salida.
x	Crimen a mostrar.

Devuelve

Referencia al flujo de salida.

5.5. Referencia del Archivo documentacion.dox

5.6. Referencia del Archivo fecha.h

Declaración de la clase fecha.

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

Clases

- class fecha

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

Funciones

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

  Sobrecarga del operador << para los flujos de salida de la clase.

5.6.1. Descripción detallada

Declaración de la clase fecha.

Autor

Alejandro Campoy Nieves
David Criado Ramón

5.6.2. Documentación de las funciones

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

Sobrecarga del operador << para los flujos de salida de la clase.

Parámetros

os	como flujo de entrada dado por referencia para indicar las operaciones a realizar.
f	como fecha dado por referencia a partir de la cual se mete sus datos en el flujo.

Devuelve

Nos devuelve dicho flujo pasado como primer parámetro.

Sobrecarga del operador << para los flujos de salida de la clase.

Parámetros

os	como flujo de entrada dado por referencia para indicar las operaciones a realizar.
f	como fecha dado por referencia a partir de la cual se mete sus datos en el flujo.

Devuelve

Nos devuelve dicho flujo pasado como primer parámetro.

5.7. Referencia del Archivo fecha.hxx

Implementación de la clase fecha.

Funciones

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

  Sobrecarga del operador << para los flujos de entrada de la clase.

5.7.1. Descripción detallada

Implementación de la clase fecha.

Autor

Alejandro Campoy Nieves
David Criado Ramón

5.7.2. Documentación de las funciones

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

Sobrecarga del operador << para los flujos de entrada de la clase.

Sobrecarga del operador << para los flujos de salida de la clase.

Parámetros

os	como flujo de entrada dado por referencia para indicar las operaciones a realizar.
f	como fecha dado por referencia a partir de la cual se mete sus datos en el flujo.

Devuelve

Nos devuelve dicho flujo pasado como primer parámetro.

5.8. Referencia del Archivo principal.cpp

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

Funciones

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

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

5.8.1. Documentación de las funciones

5.8.1.1. 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

5.8.1.2. int main ( )

Índice alfabético

arrest
  crimen, 20

case_number
  crimen, 20
cheq_rep
  conjunto, 10
conjunto, 7
  cheq_rep, 10
  conjunto, 9
  empty, 10
  entrada, 9
  erase, 10
  find, 10
  findDESCR, 11
  findIUCR, 11
  insert, 12
  operator<<, 12
  operator=, 12
  size, 12
  size_type, 9
  vc, 12
conjunto.h, 24
  operator<<, 25
conjunto.hxx, 25
  operator<<, 25
crimen, 13
  arrest, 20
  case_number, 20
  crimen, 15
  date, 20
  description, 20
  domestic, 20
  getArrest, 15
  getCaseNumber, 15
  getDESCR, 15
  getDate, 15
  getDomestic, 15
  getID, 15
  getIUCR, 15
  getLatitude, 16
  getLocation, 16
  getLongitude, 16
  getPrimaryType, 16
  ID, 20
  IUCR, 20
  latitude, 20
  location, 20
  longitude, 20
  operator<, 16
  operator<<, 18
  operator=, 16, 17
  operator==, 17
  primary_type, 20
  setArrest, 17
  setCaseNumber, 17
  setDESCR, 17
  setDate, 17
  setDomestic, 18
  setID, 18
  setIUCR, 18
  setLatitude, 18
  setLocation, 18
  setLongitude, 18
  setPrimaryType, 18
crimen.h, 26
  operator<<, 26
crimen.hxx, 26
  operator<<, 27

date
  crimen, 20
description
  crimen, 20
documentacion.dox, 27
domestic
  crimen, 20

empty
  conjunto, 10
entrada
  conjunto, 9
erase
  conjunto, 10

fecha, 20
  fecha, 21, 22
  hour, 24
  mday, 24
  min, 24
  mon, 24
  operator<, 22
  operator<<, 24
  operator<=, 22
  operator>, 23
  operator>=, 23
  operator=, 22, 23
  operator==, 23
  sec, 24
  toString, 23
  year, 24
fecha.h, 27
  operator<<, 28
fecha.hxx, 29
  operator<<, 29
find
  conjunto, 10
findDESCR
  conjunto, 11
findIUCR
  conjunto, 11

getArrest
crimen, 15
getCaseNumber
crimen, 15
getDESCR
crimen, 15
getDate
crimen, 15
getDomestic
crimen, 15
getID
crimen, 15
getIUCR
crimen, 15
getLatitude
crimen, 16
getLocation
crimen, 16
getLongitude
crimen, 16
getPrimaryType
crimen, 16

hour
fecha, 24

ID
crimen, 20
IUCR
crimen, 20
insert
conjunto, 12

latitude
crimen, 20
load
principal.cpp, 30
location
crimen, 20
longitude
crimen, 20

main
principal.cpp, 30
mday
fecha, 24
min
fecha, 24
mon
fecha, 24

operator<
crimen, 16
fecha, 22
operator<<
conjunto, 12
conjunto.h, 25
conjunto.hxx, 25
crimen, 18
crimen.h, 26
crimen.hxx, 27
fecha, 24
fecha.h, 28
fecha.hxx, 29
operator<=
fecha, 22
operator>
fecha, 23
operator>=
fecha, 23
operator=
conjunto, 12
crimen, 16, 17
fecha, 22, 23
operator==
crimen, 17
fecha, 23

primary_type
crimen, 20
principal.cpp, 30
load, 30
main, 30

sec
fecha, 24
setArrest
crimen, 17
setCaseNumber
crimen, 17
setDESCR
crimen, 17
setDate
crimen, 17
setDomestic
crimen, 18
setID
crimen, 18
setIUCR
crimen, 18
setLatitude
crimen, 18
setLocation
crimen, 18
setLongitude
crimen, 18
setPrimaryType
crimen, 18
size
conjunto, 12
size_type
conjunto, 9

toString
fecha, 23

vc
conjunto, 12

year
    fecha, 24