Práctica Templates

Generated by Doxygen 1.8.12

Contents

1 PRACTICA TEMPLATE **1**

1.1 Introducción 1

1.2 Uso de templates 2

1.3 functor 3

1.4 Generalizando el conjunto. 3

1.4.1 Insert 4

1.4.2 SE PIDE 5

2 Todo List **7**

3 Class Index **9**

3.1 Class List 9

4 Class Documentation **11**

4.1 conjunto< T, CMP > Class Template Reference 11

4.1.1 Detailed Description 12

4.1.2 Constructor & Destructor Documentation 13

4.1.2.1 conjunto() 13

4.1.3 Member Function Documentation 13

4.1.3.1 begin() 13

4.1.3.2 cbegin() 13

4.1.3.3 cend() 14

4.1.3.4 clear() 14

4.1.3.5 count() 14

4.1.3.6 empty() 15

4.1.3.7 end() . . . 15
4.1.3.8 erase() . . . 15
4.1.3.9 find() . . . 16
4.1.3.10 getVM() . . . 16
4.1.3.11 insert() . . . 16
4.1.3.12 lower_bound() . . . 17
4.1.3.13 operator=() . . . 17
4.1.3.14 size() . . . 18
4.1.3.15 upper_bound() . . . 18
4.2 CrecienteChrPos Class Reference . . . 18
4.3 CrecienteEnf Class Reference . . . 19
4.4 CrecienteID Class Reference . . . 19
4.5 DecrecienteChrPos Class Reference . . . 19
4.6 DecrecienteEnf Class Reference . . . 19
4.7 DecrecienteID Class Reference . . . 20
4.8 enfermedad Class Reference . . . 20
4.8.1 Detailed Description . . . 21
4.8.2 Constructor & Destructor Documentation . . . 21
4.8.2.1 enfermedad() . . . 21
4.8.3 Member Function Documentation . . . 21
4.8.3.1 getDatabase() . . . 21
4.8.3.2 getID() . . . 21
4.8.3.3 getName() . . . 22
4.8.3.4 imprime_Enf() . . . 22
4.8.3.5 nameContains() . . . 22
4.8.3.6 operator"!=() . . . 22
4.8.3.7 operator<() . . . 23
4.8.3.8 operator=() . . . 23
4.8.3.9 operator==() . . . 23
4.8.3.10 setDatabase() . . . 23

4.8.3.11 setID() . . . 24
4.8.3.12 setName() . . . 24
4.8.3.13 toString() . . . 24
4.9 mutacion Class Reference . . . 25
4.9.1 Detailed Description . . . 26
4.9.2 Constructor & Destructor Documentation . . . 26
4.9.2.1 mutacion() [1/2] . . . 26
4.9.2.2 mutacion() [2/2] . . . 26
4.9.3 Member Function Documentation . . . 27
4.9.3.1 getCaf() . . . 27
4.9.3.2 getChr() . . . 27
4.9.3.3 getClnsig() . . . 27
4.9.3.4 getCommon() . . . 27
4.9.3.5 getEnfermedades() . . . 28
4.9.3.6 getGenes() . . . 28
4.9.3.7 getID() . . . 28
4.9.3.8 getPos() . . . 28
4.9.3.9 getRef_alt() . . . 28
4.9.3.10 imprime_Caf() . . . 29
4.9.3.11 imprime_Enfermedades() . . . 29
4.9.3.12 imprime_Genes() . . . 29
4.9.3.13 imprime_Ref() . . . 29
4.9.3.14 operator<() . . . 29
4.9.3.15 operator=() . . . 30
4.9.3.16 operator==() . . . 30
4.9.3.17 setCaf() . . . 30
4.9.3.18 setChr() . . . 31
4.9.3.19 setClnsig() . . . 31
4.9.3.20 setCommon() . . . 31
4.9.3.21 setEnfermedades() . . . 31
4.9.3.22 setGenes() . . . 32
4.9.3.23 setID() . . . 32
4.9.3.24 setPos() . . . 32
4.9.3.25 setRef_alt() . . . 32

Index **35**

Chapter 1

PRACTICA TEMPLATE

Version

v2

Authors

Jesús Jiménez Sánchez, Jaime Frías Funes y Nazaret Ruiz Jaldo

1.1 Introducción

En la práctica anterior se os pidió la implementación del tipo conjunto de mutaciones. En esta práctica el objetivo es seguir avanzando en el uso de las estructuras de datos, particularmente mediante el uso de plantillas (templates) para la definición de tipos de datos genéricos.

Nuestro objetivo es dotar al TDA conjunto de la capacidad de almacenar elementos de cualquier tipo. Teniendo en cuenta que los elementos se almacenan de forma ordenada, es necesario que el tipo de dato con el que se instancia el conjunto tenga definido una relación de preorden total, R, esto es:

- Para todo x,y: xRy || yRx
- Para todo x,y,z: Si xRy && yRz entonces xRz

Por tanto R es una relación binaria que toma como entrada dos elementos del mismo tipo y como salida nos devuelve un booleano. Ejemplos de este tipo de relaciones son el operator$<$ (o el operator$>$), ya definidos sobre la clase mutación y enfermedad

Tanto el tipo de dato con el que particularizar el conjunto como el criterio de ordenación serán proporcionados a la hora de definir un conjunto.

```
template <typename T, typename CMP> class conjunto;
```

en la plantilla T hace referencia al tipo de dato, y CMP hace referencia al criterio de comparación interno (functor).

En la clase se declarará un objeto tipo CMP, que llamaremos comp. Así, la expresion comp(a,b) devuelve true si se considera que a precede b en la relación de preorden. Esta relación será utilizada por el "conjunto" para:

- decidir cuando un elemento precede a otro en el contenedor,
- pero también a la hora de determinar cuando dos elementos son equivalentes: para determinar cuando dos elementos serán considerados iguales con respecto a la relacion tendremos en cuenta que

  ```
  Si (!comp(a,b) && !comp(b,a))  entonces necesariamente a==b.
  ```

1.2 Uso de templates

Hasta ahora, en un conjunto de mutaciones los elementos se encuentran almacenadas en orden siguiendo el criterio de comparación dado por el operador< (primero se compara cromosoma y en caso de empate se compara la posición). Sin embargo nos podríamos plantear otros conjuntos de elementos. Así, podríamos tener

```
#include "conjunto.h"
...
// declaracion de tipos básicos:

conjunto<mutacion,less<mutacion > > Xl;  // elementos ordenados en orden creciente  (operator<
       sobre mutacion)
conjunto<mutacion,greater<mutacion > > Xg;  // elementos ordenados en
       orden creciente  (operator> sobre mutacion)

conjunto<enfermedad,greater<enfermedad> > Yg; // elementos ordenados
       en orden decreciente (operator> sobre enfermedad)

// declaración de tipos más complejos:

conjunto<mutacion,less<mutacion> >::iterator itl;
conjunto<enfermedad,greater<enfermedad> >::iterator itg;


...
```

Hay que notar que en este ejemplo Xl y Xg representan a tipos distintos, esto es un conjunto ordenado en orden creciente NO SERÁ del mismo tipo que un conjunto ordenado en orden decreciente. De igual forma, itl e itg tampoco serán variables del mismo tipo, por lo que no podríamos hacer entre otras cosas asignaciones como

```
Xl=Yg; // ERROR  un conjunto ordenado en orden descendente no puede ser asignado a un conjunto ordenado en
       orden descendente
...
itg=itl; // igual que en el caso anterior, apuntan a elementos distintos
```

En este caso, para realizar la práctica, el alumno deberá modificar tanto el fichero de especificación, conjunto.h, de forma que la propia especificación indique que trabaja con parámetros plantilla, como los ficheros de implementación (.hxx) de la clase conjunto.

De igual forma se debe modificar el fichero principal.cpp de manera que se demuestre el correcto comportamiento del conjunto cuando se instancia bajo distintos tipos de datos y distintos criterios de ordenación, en concreto debemos asegurarnos que utilizamos los siguientes criterios de ordenación:

- conjunto de mutacion creciente por cromosoma/posicion
- conjunto de mutacion decreciente por cromosoma/posicion
- conjunto de mutacion creciente por id
- conjunto de mutacion decreciente por id
- conjunto de enfermedad por orden creciente
- conjunto de enfermedad por orden decreciente

Para los dos primeros casos, y teniendo en cuenta que tenemos sobrecargado los operadores relacionales para mutación, es suficiente con utilizar las clases genéricas less<T> y greater<T> definidas en functional (#include <functional>). Sin embargo, para el resto de casos del conjunto de mutaciones debemos implementar los functores que nos permitan realizar la correcta comparación entre mutaciones.

1.3 functor

Para realizar dichas comparaciones utilizaremos una herramienta potente de C++: un functor (objeto función). Un functor es una clase en C++ que actua como una función. Un functor puede ser llamado con una sintaxis familiar a la de las funciones en C++, pudiendo devolver valores y aceptar parámetros como una función normal.

Por ejemplo, si queremos crear un functor que compare dos mutaciones teniendo en cuenta el orden de ID, podríamos hacer

```
mutacion x,y;
...
ComparacionPorID miFunctor;
cout << miFunctor(x,y) << endl;
```

Aunque miFunctor es un objeto, en la llamada miFunctor(x,y) la tratamos como si estuviésemos invocando a una función tomando x e y como parámetros.

Para crear dicho functor, creamos un objeto que sobrecarga el operador() como sigue

```
class ComparacionPorID {
 public:
   bool operator()(const mutacion &a, const mutacion &b) {
     return (a.getID() < b.getID()); // devuelve verdadero si el ID de a precede al ID de b
 }
};
```

1.4 Generalizando el conjunto.

Para poder extender nuestro conjunto hemos de dotarlo de la capacidad de poder definir el criterio de ordenación. Para ello vamos a considerar un caso simplificado (que no se corresponde exactamente con lo que se pide en la práctica) donde ilustraremos su uso

```
template <typename T, typename CMP>
class conjunto {
  public:
    ....
    pair<typename conjunto<T,CMP>::iterator,bool> insert( const T & c);

  private:
    vector<T> vdatos; //donde se almacenan los datos
    CMP comp;
};
```

Como hemos dicho, el nombre del tipo ahora es conjunto<T,CMP> y no conjunto. Distintas particularizaciones dan lugar a tipos también distintos. Ahora, en el fichero conjunto.hxx debemos de implementar cada uno de los métodos, recordemos que cada uno de ellos pertenece a la clase conjunto<T,CMP> y por tanto se implementa considerando

```
tipoRetorno conjunto<T,CMP>::nombreMetodo( parametros ...)
```

Pasamos a ver la implementación de los métodos:

1.4.1 Insert

El método insert asume como prerequisito que el conjunto está ordenado según el criterio dado por CMP, y por tanto debe asegurar que tras insertar una nueva entrada dicho conjunto siga ordenado. Por ejemplo, podríamos hacer algo del tipo:

```
pair<typename conjunto<T,CMP>::iterator,bool>
                              conjunto<T,CMP>::insert( const T & c){

      pair<typename conjunto<T,CMP>::iterator,bool> salida;

      bool fin = false;
      for (auto it = vdatos.begin(); it!=vdatos.end() && !fin; ){
        if (comp(*it,c) ) it++;
        else if (!comp(*it,c) && !comp(c,*it)){ // equivalentes segun CMP
            salida.first = vdatos.end();
            salida.second = false;
            fin = true;
        }else {
            salida.first = vdatos.insert(it,c);
            salida.second = fin = true;
        }
      } // del for
      if (!fin){
        salida.first = vdatos.insert(vdatos.end(),c);
        salida.second = true;
      }
      return salida;
}
```

En este caso comp(∗it,c) hace referencia a una comparación genérica entre elementos de tipo T definida por la relación de orden con la que se haya particularizado el conjunto. Así si hemos definido

```
conjunto<mutacion,ComparacionPorID> cID;
```

en este caso comp es un objeto de la clase ComparacionPorID, y mediante la llamada comp(.,.) lo que estamos haciendo es llamar a la "función" que me compara dos mutaciones teniendo en cuenta su ID.

Finalmente, debemos tener cuidado a la hora de realizar comparaciones y la semántica de las mismas, como se muestra en el caso en que comparamos cuando dos elementos son "iguales". Así, si en lugar de realizar la comparación

```
if (!comp(*it,c) && !comp(c,*it)){ // equivalentes segun CMP
}
```

hubiésemos utilizado

```
If (*it==c) { //igualdad
```

estaríamos haciendo la llamada a la comparación de igualdad entre mutaciones (definida mediante la comparación de cromosoma/posicion) por lo que podría funcionar correctamente el método cuando particularizamos con

```
Conjunto<mutacion,less<mutacion> > X;
```

Sin embargo, si el conjunto está definido como conjunto<mutacion,ComparacionPorID>, utilizar el mismo código para realizar la búsqueda no funcionaría correctamente: los elementos están ordenados en orden creciente de ID y la comparación de igualdad se hace por cromosoma/mutacion.

1.4.2 SE PIDE

Por tanto, se pide la implementación de los métodos de la clase conjunto genérico y su prueba de funcionamiento correcto en los supuestos planteados anteriormente

- conjunto de mutacion creciente por cromosoma/posicion
- conjunto de mutacion decreciente por cromosoma/posicion
- conjunto de mutacion creciente por id
- conjunto de mutacion decreciente por id
- conjunto de enfermedad por orden creciente
- conjunto de enfermedad por orden decreciente

Dicha entrega se debe realizar antes del 18 de Noviembre a las 23:59 horas.

Chapter 2

Todo List

Class conjunto< T, CMP >

Implementa esta clase siguiendo la especificación asociada

Class enfermedad

Implementa esta clase, junto con su documentación asociada

Class mutacion

Implementa esta clase, junto con su documentación asociada

Chapter 3

Class Index

3.1 Class List

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

conjunto< T, CMP >
: Clase conjunto 11
CrecienteChrPos 18
CrecienteEnf 19
CrecienteID 19
DecrecienteChrPos 19
DecrecienteEnf 19
DecrecienteID 20
enfermedad
: Clase enfermedad, asociada al TDA enfermedad 20
mutacion
: Clase mutacion, asociada a la definición de una mutación/SNP 25

Chapter 4

Class Documentation

4.1 conjunto< T, CMP > Class Template Reference

Clase conjunto.

`#include <conjunto.h>`

Public Types

- typedef T **value_type**
- typedef unsigned int **size_type**
- typedef vector< value_type >::iterator **iterator**
- typedef vector< value_type >::const_iterator **const_iterator**

Public Member Functions

- conjunto ()
  constructor primitivo.
- conjunto (const conjunto< T, CMP > &d)
  constructor de copia
- iterator find (const value_type &s)
  busca una entrada en el conjunto
- const_iterator **find** (const value_type &s) const
- size_type count (const value_type &e) const
  cuenta cuantas entradas coinciden con los parámetros dados.
- pair< iterator, bool > insert (const value_type &val)
  Inserta una entrada en el conjunto.
- pair< iterator, bool > **insert** (value_type &val)
- iterator erase (const iterator position)
  Borra una entrada en el conjunto . Busca la entrada y si la encuentra la borra.
- size_type **erase** (const value_type &val)
- void clear ()
  Borra todas las entradas del conjunto, dejandolo vacio.
- size_type size () const
  numero de entradas en el conjunto

- bool empty () const

  Chequea si el conjunto esta vacio (size()==0)
- conjunto & operator= (const conjunto &org)

  operador de asignación
- vector< value_type > & **operator=** (vector< value_type > org)
- conjunto::iterator begin ()

  begin del conjunto
- conjunto::iterator end ()

  end del conjunto
- conjunto::const_iterator cbegin () const

  begin del conjunto
- conjunto::const_iterator cend () const

  end del conjunto
- iterator lower_bound (value_type &val)

  busca primer elemento que no está por debajo ('antes', '<') de los parámetros dados.
- const_iterator **lower_bound** (const value_type &val) const
- iterator upper_bound (const value_type &val)

  busca primer elemento por encima ('después', '>') de los parámetros dados.
- const_iterator **upper_bound** (const value_type &val) const
- const vector< mutacion > & getVM ()

  Devuelve vm.

4.1.1 Detailed Description

template<typename T, typename CMP>
class conjunto< T, CMP >

Clase conjunto.

conjunto<T,CMP>::conjunto, find, size, insert, erase,

Tipos conjunto<T,CMP>::value_type, conjunto<T,CMP>::size_type, conjunto<T,CMP>::iterator, conjunto<T,CMP>::const_iterator

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::value_type
```

que permite hacer referencia al elemento almacenados en cada una de las posiciones del conjunto, en nuestro caso mutaciones (SNPs) o enfermedades. Es requisito que el tipo conjunto::value_type tenga definida una relación de orden, CMP, y el operador de asignación, operator= .

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

Todo Implementa esta clase siguiendo la especificación asociada

4.1.2 Constructor & Destructor Documentation

4.1.2.1 conjunto()

```
template<typename T, typename CMP>
conjunto< T, CMP >::conjunto (
            const conjunto< T, CMP > & d )
```

constructor de copia

Parameters

in	*d*	conjunto a copiar

4.1.3 Member Function Documentation

4.1.3.1 begin()

```
template<typename T , typename CMP >
conjunto< T, CMP >::iterator conjunto< T, CMP >::begin ( )
```

begin del conjunto

Returns

Devuelve un iterador al primer elemento del conjunto. Si no existe devuelve end

Postcondition

no modifica el conjunto.

4.1.3.2 cbegin()

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

begin del conjunto

Returns

Devuelve un iterador constante al primer elemento del conjunto. Si no existe devuelve cend

Postcondition

no modifica el conjunto.

4.1.3.3 cend()

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

end del conjunto

Returns

Devuelve un iterador constante al final del conjunto (posicion siguiente al ultimo.

Postcondition

no modifica el conjunto.

4.1.3.4 clear()

```
template<typename T , typename CMP >
void conjunto< T, CMP >::clear ( )
```

Borra todas las entradas del conjunto, dejandolo vacio.

Postcondition

El conjunto se modifica, quedando vacio.

4.1.3.5 count()

```
template<typename T, typename CMP>
conjunto< T, CMP >::size_type conjunto< T, CMP >::count (
            const value_type & e ) const
```

cuenta cuantas entradas coinciden con los parámetros dados.

Parameters

in	*e*	entrada.

Returns

Como el conjunto no puede tener entradas repetidas, devuelve 1 (si se encuentra la entrada) o 0 (si no se encuentra).

Postcondition

no modifica el conjunto.

4.1.3.6 empty()

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

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

Postcondition

No se modifica el conjunto.

4.1.3.7 end()

```
template<typename T , typename CMP >
conjunto< T, CMP >::iterator conjunto< T, CMP >::end ( )
```

end del conjunto

Returns

Devuelve un iterador al final del conjunto (posicion siguiente al ultimo).

Postcondition

no modifica el conjunto.

4.1.3.8 erase()

```
template<typename T, typename CMP>
iterator conjunto< T, CMP >::erase (
            const iterator position )
```

Borra una entrada en el conjunto . Busca la entrada y si la encuentra la borra.

Parameters

in	*val*	entrada a borrar.
in	*position*	itarador que apunta a la entrada que queremos borrar

Returns

devuelve la posicion siguiente al elemento borrado (para la version con iterador) o el numero de elementos borrados

Postcondition

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

4.1.3.9 find()

```
template<typename T, typename CMP>
iterator conjunto< T, CMP >::find (
            const value_type & s )
```

busca una entrada en el conjunto

Parameters

in	*s*	entrada a buscar.

Returns

Si existe una entrada en el conjunto con ese valor devuelve el iterador a su posicion, en caso contrario devuelve iterador al final de conjunto

Postcondition

no modifica el conjunto.

4.1.3.10 getVM()

```
template<typename T , typename CMP >
const vector< mutacion > & conjunto< T, CMP >::getVM ( )
```

Devuelve vm.

Returns

vm vector de mutaciones

Devuelve el vector de todas las mutaciones.

4.1.3.11 insert()

```
template<typename T, typename CMP>
pair<iterator,bool> conjunto< T, CMP >::insert (
            const value_type & val )
```

Inserta una entrada en el conjunto.

Parameters

val	entrada a insertar

Returns

un par donde el segundo campo vale true si la entrada se ha podido insertar con éxito, esto es, no existe una mutación con igual valor en el conjunto. False en caso contrario. El primer campo del par devuelve un iterador al elemento insertado, o end() si no fue posible la insercion

Postcondition

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

4.1.3.12 lower_bound()

```
template<typename T, typename CMP>
iterator conjunto< T, CMP >::lower_bound (
            value_type & val )
```

busca primer elemento que no está por debajo ('antes', '<') de los parámetros dados.

Parameters

in	*val*	entrada.

Returns

Devuelve un iterador al primer elemento que cumple que "elemento<e" es falso, esto es, el primer elemento que es mayor o igual que val Si no existe devuelve end

Postcondition

no modifica el conjunto.

4.1.3.13 operator=()

```
template<typename T , typename CMP >
conjunto< T, CMP > & conjunto< T, CMP >::operator= (
            const conjunto< T, CMP > & org )
```

operador de asignación

Parameters

in	*org*	conjunto a copiar.

Returns

Crea y devuelve un conjunto duplicado exacto de org.

4.1.3.14 size()

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

numero de entradas en el conjunto

Postcondition

No se modifica el conjunto.

Returns

numero de entradas en el conjunto

4.1.3.15 upper_bound()

```
template<typename T , typename CMP >
conjunto< T, CMP >::iterator conjunto< T, CMP >::upper_bound (
            const value_type & val )
```

busca primer elemento por encima ('después', '>') de los parámetros dados.

Parameters

in	*val*	entrada. Devuelve un iterador al primer elemento que cumple que "elemento>e", esto es, el primer elemento ESTRICTAMENTE mayor que val Si no existe devuelve end

Postcondition

no modifica el conjunto.

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

- conjunto.h
- conjunto.hxx

4.2 CrecienteChrPos Class Reference

Public Member Functions

- bool **operator()** (const mutacion &a, const mutacion &b)

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

- principal.cpp

4.3 CrecienteEnf Class Reference

Public Member Functions

- bool **operator()** (const enfermedad &a, const enfermedad &b)

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

- principal.cpp

4.4 CrecienteID Class Reference

Public Member Functions

- bool **operator()** (const mutacion &a, const mutacion &b)

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

- principal.cpp

4.5 DecrecienteChrPos Class Reference

Public Member Functions

- bool **operator()** (const mutacion &a, const mutacion &b)

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

- principal.cpp

4.6 DecrecienteEnf Class Reference

Public Member Functions

- bool **operator()** (const enfermedad &a, const enfermedad &b)

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

- principal.cpp

4.7 DecrecienteID Class Reference

Public Member Functions

- bool **operator()** (const mutacion &a, const mutacion &b)

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

- principal.cpp

4.8 enfermedad Class Reference

Clase enfermedad, asociada al TDA enfermedad.

`#include <enfermedad.h>`

Public Member Functions

- enfermedad ()
  Crea una enfermedad iniciando chr y pos a 1.
- enfermedad (const string &name, const string &ID, const string &database)
  Crea una enfermedad a partir de 3 cadenas de string.
- void setName (const string &name)
  Cambia el valor de name.
- void setID (const string &ID)
  Cambia el valor de ID.
- void setDatabase (const string &database)
  Cambia el valor de database.
- string getName () const
  Devuelve el nombre.
- string getID () const
  Devuelve el ID.
- string getDatabase () const
  Devuelve database.
- enfermedad & operator= (const enfermedad &e)
  Sobrecarga del operador =.
- string toString () const
  Convierte una enfermedad a una cadena string.
- bool operator== (const enfermedad &e) const
  Sobrecarga del operador ==.
- bool operator!= (const enfermedad &e) const
  Sobrecarga del operador !=.
- bool operator< (const enfermedad &e) const
  Sobrecarga del operador <.
- bool nameContains (const string &str) const
  Devuelve si el nombre contiene el string argumentado.
- string imprime_Enf () const
  Guarda la enfermedad en un string.

4.8.1 Detailed Description

Clase enfermedad, asociada al TDA enfermedad.

enfermedad::enfermedad, ... Descripción contiene toda la información asociada a una enfermedad almacenada en la BD ClinVar-dbSNP (nombre de la enfermedad, id, BD que provee el id)

Todo Implementa esta clase, junto con su documentación asociada

4.8.2 Constructor & Destructor Documentation

4.8.2.1 enfermedad()

```
enfermedad::enfermedad (
            const string & name,
            const string & ID,
            const string & database )
```

Crea una enfermedad a partir de 3 cadenas de string.

Parameters

name	string con el nombre
ID	string con el ID
database	string con la base de datos que provee el ID

4.8.3 Member Function Documentation

4.8.3.1 getDatabase()

```
string enfermedad::getDatabase ( ) const
```

Devuelve database.

Returns

database base de datos

Devuelve la base de datos de la enfermedad.

4.8.3.2 getID()

```
string enfermedad::getID ( ) const
```

Devuelve el ID.

Returns

ID ID de la enfermedad

Devuelve el ID de la enfermedad.

4.8.3.3 getName()

```
string enfermedad::getName ( ) const
```

Devuelve el nombre.

Returns

name nombre de la enfermedad

Devuelve el nombre de la enfermedad.

4.8.3.4 imprime_Enf()

```
string enfermedad::imprime_Enf ( ) const
```

Guarda la enfermedad en un string.

Returns

string Cadena de string con la enfermedad

Convierte la enfermedad en un string.

4.8.3.5 nameContains()

```
bool enfermedad::nameContains (
            const string & str ) const
```

Devuelve si el nombre contiene el string argumentado.

Parameters

str	string a buscar en el nombre

Devuelve si está el string argumentado en el nombre de la enfermedad aunque no se trate del nombre completo. No es sensible a mayúsculas/minúsculas

4.8.3.6 operator"!=()

```
bool enfermedad::operator!= (
            const enfermedad & e ) const
```

Sobrecarga del operador !=.

Parameters

e	enfermedad a comparar

Sobrecarga el operador de asignación para ajustarlo a enfermedades.

4.8.3.7 operator<()

```
bool enfermedad::operator< (
            const enfermedad & e ) const
```

Sobrecarga del operador <.

Parameters

e	enfermedad a comparar

Sobrecarga el operador de asignación para comparar dos enfermedades en función del orden alfabético de l campo name.

4.8.3.8 operator=()

```
enfermedad & enfermedad::operator= (
            const enfermedad & e )
```

Sobrecarga del operador =.

Parameters

e	enfermedad a igualar

Sobrecarga el operador de asignación para ajustarlo a enfermedades.

4.8.3.9 operator==()

```
bool enfermedad::operator== (
            const enfermedad & e ) const
```

Sobrecarga del operador ==.

Parameters

e	enfermedad a comparar

Sobrecarga el operador de comparación para ajustarlo a una enfermedad.

4.8.3.10 setDatabase()

```
void enfermedad::setDatabase (
            const string & database )
```

Cambia el valor de database.

Parameters

database	nuevo valor de database

Cambia el valor de la base de datos de la enfermedad.

4.8.3.11 setID()

```
void enfermedad::setID (
            const string & ID )
```

Cambia el valor de ID.

Parameters

ID	nuevo valor de ID

Cambia el valor del ID de la enfermedad.

4.8.3.12 setName()

```
void enfermedad::setName (
            const string & name )
```

Cambia el valor de name.

Parameters

name	string con el nuevo nombre

Cambia el valor del nombre de la enfermedad.

4.8.3.13 toString()

```
string enfermedad::toString ( ) const
```

Convierte una enfermedad a una cadena string.

Returns

str cadena a devolver con la enfermedad

Se convierte una enfermedad a una cadena string.

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

- enfermedad.h
- enfermedad.cpp

4.9 mutacion Class Reference

Clase mutacion, asociada a la definición de una mutación/SNP.

`#include <mutacion.h>`

Public Member Functions

- mutacion ()
  Inicia una mutacion vacía.
- mutacion (const mutacion &m)
  Crea una mutación a partir de otra.
- mutacion (const string &str)
  Crea una mutación a partir de un string.
- void setID (const string &id)
  Cambia el valor de ID.
- void setChr (const string &chr)
  Cambia el valor del cromosoma.
- void setPos (const unsigned int &pos)
  Cambia el valor de la posición.
- void setRef_alt (const vector< string > &ref_alt)
  Cambia el valor del genoma básico.
- void setGenes (const vector< string > &genes)
  Cambia el valor de los genes.
- void setCommon (const bool &common)
  Cambia el valor del bool de si es común o no la mutación.
- void setCaf (const vector< float > &caf)
  Cambia el valor de la frecuencia de cada base.
- void setEnfermedades (const vector< enfermedad > &enfermedades)
  Cambia el valor del vector de enfermedades.
- void setClnsig (const vector< int > &clnsig)
  Cambia la relevancia clínica.
- string getID () const
  Devuelve el ID de la mutación.
- string getChr () const
  Devuelve el identificador del cromosoma.
- unsigned int getPos () const
  Devuelve la posición del cromosoma.
- const vector< string > & getRef_alt () const
  Devuelve el vector del genoma básico.
- const vector< string > & getGenes () const
  Devuelve el vector de los genes.
- bool getCommon () const
  Devuelve si una mutación es común o no.
- const vector< float > & getCaf () const
  Devuelve el vector de las frecuencias de cada base.
- const vector< enfermedad > & getEnfermedades () const
  Devuelve el vector de las enfermedades.
- const vector< int > & getClnsig () const
  Devuelve el vector de enteros que conforman la relevancia clínica.

- string imprime_Ref () const

  Devuelve la relevancia clínica convertida en el string hola.
- string imprime_Genes () const

  Devuelve la referencia de los genes convertida en el string hola.
- string imprime_Caf () const

  Devuelve la frecuencia de cada base.
- string imprime_Enfermedades () const

  Devuelve la las enfermedades asociadas.
- mutacion & operator= (const mutacion &m)

  Sobrecarga el operador = para ajustarlo a una mutación.
- bool operator== (const mutacion &m) const

  Sobrecarga el operador == para compararlo con una mutación.
- bool operator< (const mutacion &m) const

  ESobrecarga el operador < para comparar una mutación.

4.9.1 Detailed Description

Clase mutacion, asociada a la definición de una mutación/SNP.

mutacion::mutacion, ...

Todo Implementa esta clase, junto con su documentación asociada

4.9.2 Constructor & Destructor Documentation

4.9.2.1 mutacion() [1/2]

```
mutacion::mutacion (
            const mutacion & m )
```

Crea una mutación a partir de otra.

Parameters

m	mutación a copiar

Copia los valores de la mutación pasada como argumento y crea una nueva a partir de ellos.

Suponemos que la mutación que nos pasan tiene bien los valores, es decir pos>1

4.9.2.2 mutacion() [2/2]

```
mutacion::mutacion (
            const string & str )
```

Crea una mutación a partir de un string.

Parameters

str	cadena string con los datos miembro de la clase

Se le pasa un string como argumento y se recorre este guardando los correspondientes datos miembro.

4.9.3 Member Function Documentation

4.9.3.1 getCaf()

```
const vector< float > & mutacion::getCaf ( ) const
```

Devuelve el vector de las frecuencias de cada base.

Returns

caf vector de las frecuencias

4.9.3.2 getChr()

```
string mutacion::getChr ( ) const
```

Devuelve el identificador del cromosoma.

Returns

chr identificador del cromosoma

4.9.3.3 getClnsig()

```
const vector< int > & mutacion::getClnsig ( ) const
```

Devuelve el vector de enteros que conforman la relevancia clínica.

Returns

clnsig vector de enteros con la relevancia clínica.

4.9.3.4 getCommon()

```
bool mutacion::getCommon ( ) const
```

Devuelve si una mutación es común o no.

Returns

common bool de si es común o no

4.9.3.5 getEnfermedades()

```
const vector< enfermedad > & mutacion::getEnfermedades ( ) const
```

Devuelve el vector de las enfermedades.

Returns

enfermedad vector de las enfermedades

4.9.3.6 getGenes()

```
const vector< string > & mutacion::getGenes ( ) const
```

Devuelve el vector de los genes.

Returns

genes vector de los genes

4.9.3.7 getID()

```
string mutacion::getID ( ) const
```

Devuelve el ID de la mutación.

Returns

ID valor del id de la mutación

4.9.3.8 getPos()

```
unsigned int mutacion::getPos ( ) const
```

Devuelve la posición del cromosoma.

Returns

pos posición del cromosoma

4.9.3.9 getRef_alt()

```
const vector< string > & mutacion::getRef_alt ( ) const
```

Devuelve el vector del genoma básico.

Returns

ref_alt vector del genoma básico

4.9.3.10 imprime_Caf()

```
string mutacion::imprime_Caf ( ) const
```

Devuelve la frecuencia de cada base.

Returns

hola string con la frecuencia

4.9.3.11 imprime_Enfermedades()

```
string mutacion::imprime_Enfermedades ( ) const
```

Devuelve la las enfermedades asociadas.

Returns

hola string con las enfermedades

4.9.3.12 imprime_Genes()

```
string mutacion::imprime_Genes ( ) const
```

Devuelve la referencia de los genes convertida en el string hola.

Returns

hola string con la referencia

4.9.3.13 imprime_Ref()

```
string mutacion::imprime_Ref ( ) const
```

Devuelve la relevancia clínica convertida en el string hola.

Returns

hola string con la relevancia clínica.

4.9.3.14 operator<()

```
bool mutacion::operator< (
            const mutacion & m ) const
```

ESobrecarga el operador $<$ para comparar una mutación.

Parameters

m	mutación a comparar

Returns

menor

4.9.3.15 operator=()

```
mutacion & mutacion::operator= (
            const mutacion & m )
```

Sobrecarga el operador = para ajustarlo a una mutación.

Parameters

m	mutación a la que se iguala

Returns

∗this

4.9.3.16 operator==()

```
bool mutacion::operator== (
            const mutacion & m ) const
```

Sobrecarga el operador == para compararlo con una mutación.

Parameters

m	mutación a comparar

Returns

iguales

4.9.3.17 setCaf()

```
void mutacion::setCaf (
            const vector< float > & caf )
```

Cambia el valor de la frecuencia de cada base.

Parameters

caf	nuevo vector con frecuencias

4.9.3.18 setChr()

```
void mutacion::setChr (
            const string & chr )
```

Cambia el valor del cromosoma.

Parameters

chr	nuevo cromosoma

4.9.3.19 setClnsig()

```
void mutacion::setClnsig (
            const vector< int > & clnsig )
```

Cambia la relevancia clínica.

Parameters

clnsig	nuevo vector de enteros

4.9.3.20 setCommon()

```
void mutacion::setCommon (
            const bool & common )
```

Cambia el valor del bool de si es común o no la mutación.

Parameters

common	nuevo bool

4.9.3.21 setEnfermedades()

```
void mutacion::setEnfermedades (
            const vector< enfermedad > & enfermedades )
```

Cambia el valor del vector de enfermedades.

Parameters

enfermedades	nuevo vector con enfermedades

4.9.3.22 setGenes()

```
void mutacion::setGenes (
            const vector< string > & genes )
```

Cambia el valor de los genes.

Parameters

genes	vector con los genes

4.9.3.23 setID()

```
void mutacion::setID (
            const string & id )
```

Cambia el valor de ID.

Parameters

id	id de la mutación

4.9.3.24 setPos()

```
void mutacion::setPos (
            const unsigned int & pos )
```

Cambia el valor de la posición.

Parameters

pos	posicion nueva

4.9.3.25 setRef_alt()

```
void mutacion::setRef_alt (
            const vector< string > & ref_alt )
```

Cambia el valor del genoma básico.

Parameters

ref_alt	vector de string que contiene el genoma tipo

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

- mutacion.h
- mutacion.cpp

Index

begin
    conjunto, 13

cbegin
    conjunto, 13
cend
    conjunto, 13
clear
    conjunto, 14
conjunto
    begin, 13
    cbegin, 13
    cend, 13
    clear, 14
    conjunto, 13
    count, 14
    empty, 14
    end, 15
    erase, 15
    find, 15
    getVM, 16
    insert, 16
    lower_bound, 17
    operator=, 17
    size, 17
    upper_bound, 18
conjunto< T, CMP >, 11
count
    conjunto, 14
CrecienteChrPos, 18
CrecienteEnf, 19
CrecienteID, 19

DecrecienteChrPos, 19
DecrecienteEnf, 19
DecrecienteID, 20

empty
    conjunto, 14
end
    conjunto, 15
enfermedad, 20
    enfermedad, 21
    getDatabase, 21
    getID, 21
    getName, 21
    imprime_Enf, 22
    nameContains, 22
    operator!=, 22
    operator<, 23
    operator=, 23
    operator==, 23
    setDatabase, 23
    setID, 24
    setName, 24
    toString, 24
erase
    conjunto, 15

find
    conjunto, 15

getCaf
    mutacion, 27
getChr
    mutacion, 27
getClnsig
    mutacion, 27
getCommon
    mutacion, 27
getDatabase
    enfermedad, 21
getEnfermedades
    mutacion, 27
getGenes
    mutacion, 28
getID
    enfermedad, 21
    mutacion, 28
getName
    enfermedad, 21
getPos
    mutacion, 28
getRef_alt
    mutacion, 28
getVM
    conjunto, 16

imprime_Caf
    mutacion, 28
imprime_Enf
    enfermedad, 22
imprime_Enfermedades
    mutacion, 29
imprime_Genes
    mutacion, 29
imprime_Ref
    mutacion, 29
insert
    conjunto, 16

lower_bound
conjunto, 17

mutacion, 25
getCaf, 27
getChr, 27
getClnsig, 27
getCommon, 27
getEnfermedades, 27
getGenes, 28
getID, 28
getPos, 28
getRef_alt, 28
imprime_Caf, 28
imprime_Enfermedades, 29
imprime_Genes, 29
imprime_Ref, 29
mutacion, 26
operator<, 29
operator=, 30
operator==, 30
setCaf, 30
setChr, 31
setClnsig, 31
setCommon, 31
setEnfermedades, 31
setGenes, 32
setID, 32
setPos, 32
setRef_alt, 32

nameContains
enfermedad, 22

operator!=
enfermedad, 22
operator<
enfermedad, 23
mutacion, 29
operator=
conjunto, 17
enfermedad, 23
mutacion, 30
operator==
enfermedad, 23
mutacion, 30

setCaf
mutacion, 30
setChr
mutacion, 31
setClnsig
mutacion, 31
setCommon
mutacion, 31
setDatabase
enfermedad, 23
setEnfermedades
mutacion, 31
setGenes
mutacion, 32
setID
enfermedad, 24
mutacion, 32
setName
enfermedad, 24
setPos
mutacion, 32
setRef_alt
mutacion, 32
size
conjunto, 17

toString
enfermedad, 24

upper_bound
conjunto, 18