

TDA Mutación y TDA Enfermedad

V1

Generado por Doxygen 1.8.11

Índice

1 Documentación Práctica	2
1.1 Introducción	2
1.1.1 Contexto	3
1.1.2 Conjunto de Datos	4
1.2 TDA enfermedad	5
1.3 Mutación	5
1.4 "Se Entrega / Se Pide"	6
1.4.1 Se entrega	6
1.4.2 Se Pide	7
1.5 "Fecha Límite de Entrega"	7
2 Lista de tareas pendientes	7
3 Índice de clases	7
3.1 Lista de clases	7
4 Índice de archivos	8
4.1 Lista de archivos	8
5 Documentación de las clases	8
5.1 Referencia de la Clase enfermedad	8
5.1.1 Descripción detallada	9
5.1.2 Documentación del constructor y destructor	9
5.1.3 Documentación de las funciones miembro	9
5.1.4 Documentación de los datos miembro	12
5.2 Referencia de la Clase mutacion	12
5.2.1 Descripción detallada	13
5.2.2 Documentación del constructor y destructor	14
5.2.3 Documentación de las funciones miembro	14
5.2.4 Documentación de los datos miembro	15

6 Documentación de archivos	15
6.1 Referencia del Archivo documentacion.dox	15
6.2 Referencia del Archivo enfermedad.h	15
6.2.1 Documentación de las funciones	15
6.3 Referencia del Archivo enfermedad.hxx	16
6.3.1 Documentación de las funciones	16
6.4 Referencia del Archivo mutacion.h	17
6.4.1 Documentación de las funciones	17
6.5 Referencia del Archivo principal.cpp	17
6.5.1 Documentación de las funciones	17
Índice	19

1. Documentación Práctica

Versión

v1

Autor

Carlos Cano y Juan F. Huete

1.1. Introducción

En esta practica se pretende avanzar en el uso de las estructuras de datos mediante el diseño de distintos tipos de datos para manejar la información asociada a una base de datos de mutaciones del genoma humano con relevancia clínica (ClinVar-dbsnp).

1.1.1. Contexto

El ácido desoxirribonucleico, abreviado como ADN, es un ácido nucleico que contiene las instrucciones genéticas usadas en el desarrollo y funcionamiento de todos los organismos vivos conocidos y algunos virus, y es responsable de su transmisión hereditaria. En ocasiones, se compara al ADN con un programa de ordenador, ya que contiene las instrucciones necesarias para construir otros componentes de las células, como las proteínas y las moléculas de ARN, que son las responsables del funcionamiento celular. Los segmentos de ADN que llevan esta información genética son llamados genes.

Podemos representar el ADN como una secuencia de nucleótidos (Adenina A, Timina T, Citosina C, Guanina G). La disposición secuencial de estas cuatro bases a lo largo de la cadena es la que codifica la información genética. Por ejemplo, podemos representar una pequeña cadena de ADN como: "ACCCAGTCGGATTT".

En los organismos vivos, el ADN no suele existir como una molécula individual, sino como una pareja de moléculas que se enroscan sobre sí mismas formando una especie de escalera de caracol, denominada doble hélice. Esta estructura se sustenta en la complementariedad de sus bases (Citosina-Guanina y Adenina-Timina). Al ser las bases complementarias, podemos representar el ADN sin perder información especificando sólo una de sus cadenas.

El genoma humano es una secuencia de ADN contenida en 23 pares de cromosomas en el núcleo de cada célula humana (de cada pareja de cromosomas, uno es heredado del padre y otro de la madre). Los cromosomas 1 a 22 se numeran en orden creciente de tamaño. La pareja de cromosomas 23, también llamados cromosomas sexuales, se compone de un cromosoma X (de la madre) y uno X o Y (del padre).

El tamaño total del genoma humano haploide (es decir, considerando sólo uno de cada pareja de cromosomas) es de aproximadamente 3200 millones de pares de bases de ADN. Dado que una base se representa con un Byte ('A', 'C', 'G', 'T'), el tamaño aproximado de la secuencia completa de un genoma humano haploide es de 3 GBytes.

Dos seres humanos del mismo sexo comparten un porcentaje muy elevado (99,5 %) de su secuencia de ADN, pero estas secuencias no son idénticas. Estos millones de pequeñas variaciones en el genoma, junto con la influencia de factores del medio, son los responsables de que exhibamos distintos fenotipos, es decir, distintos rasgos físicos y conductales. Una variación en el genoma, por sustitución, inserción o delección de bases, se llama mutación o polimorfismo, y la principal fuente de variabilidad entre dos genomas humanos es el polimorfismo de una sola base (Single Nucleotide Polimorphism, SNP).

Un SNP es, por tanto, un cambio de una base en una misma posición entre dos genomas humanos. Un SNP suele representarse indicando el número de cromosoma en el que se localiza el cambio, la posición dentro del cromosoma, y el cambio de base respecto al genoma humano de referencia (el primer genoma humano para el que se conoce la secuencia, que se terminó de secuenciar por primera vez en 2001). Por ejemplo, el siguiente SNP indica un cambio en la posición 1014143 del cromosoma 1, que en el genoma humano de referencia presenta una 'C' y en otros genomas presenta una 'T':

1 1014143 C T

Los SNP constituyen hasta el 90 % de todas las variaciones genómicas humanas. Estas variaciones en la secuencia del ADN pueden afectar a la respuesta de los individuos a enfermedades, bacterias, virus, productos químicos, fármacos, etc.. De este modo, su estudio es de gran utilidad en la denominada Medicina Personalizada o Medicina de Precisión: el desarrollo de métodos de prevención, diagnóstico y tratamiento (fármacos) de forma individualizada para cada paciente.

Los estudios genéticos personalizados se basan en décadas de descubrimientos científicos publicados en la literatura especializada que muestran evidencia de que la presencia de un determinado SNP en el genoma de un individuo puede hacerle propenso a padecer una cierta enfermedad. La base de datos ClinVar-dbSNP recoge esta información.

Para leer más sobre el contexto del problema:

- https://es.wikipedia.org/wiki/Ácido_desoxirribonucleico
- https://es.wikipedia.org/wiki/Genoma_humano
- https://es.wikipedia.org/wiki/Polimorfismo_de_nucleótido_único

1.1.2. Conjunto de Datos

El conjunto de datos con el que trabajaremos es la base de datos completa ClinVar-dbSNP descargada de la web del National Institute of Health (NIH) de los Estados Unidos: <https://www.ncbi.nlm.nih.gov/clinvar/>. Esta base de datos se puede obtener en formato VCF v4.0 (archivo: clinvar_20160831.vcf), que representa de forma tabular más de 130.000 mutaciones (SNPs) conocidos hasta la fecha y su relación clínica con alguna enfermedad.

El fichero comienza con una cabecera (líneas que se inician con '#') que describe cada uno de los campos de la base de datos. A partir de la línea 67 se listan las entradas de la BD, con un SNP por línea, y los campos delimitados por tabulador ('\t'). Nota: algunos campos no relevantes se han omitido en este ejemplo para facilitar su lectura (los campos omitidos se han reemplazado por [...]).

```
#CHROM POS ID REF ALT QUAL FILTER INFO
1 1014143 rs786201005 C T . . RS=786201005; [...] GENEINFO=ISG15:9636; CLNSIG=5; CLNDSDB=MedGen:OMIM;
CLNDSDBID=CN221808:616126; CLNDBN=Immunodeficiency_38_with_basal_ganglia_calcification; [...]
1 1014316 rs672601345 C CG . . RS=672601345; [...] GENEINFO=ISG15:9636; CLNSIG=5; CLNDSDB=MedGen:OMIM;
CLNDSDBID=CN221808:616126; CLNDBN=Immunodeficiency_38_with_basal_ganglia_calcification; [...]
1 1053827 rs74685771 G A,C,T . . RS=74685771; [...] GENEINFO=AGRN:375790; [...] CLNSIG=3; CLNDSDB=MedGen;
CLNDSDBID=CN169374; CLNDBN=not_specified; [...]
1 11847114 rs202102042 C T . . RS=202102042; [...] GENEINFO=NPPA:4878|NPPA-AS1:100379251; [...] CLNSIG=5;
CLNDSDB=MedGen:OMIM; CLNDSDBID=C3810401:615745; CLNDBN=Atrial_standstill_2; [...] CAF=0.9998,0.0001997;COMMON
=0
1 11847311 rs755212754 G A . . RS=755212754; [...] GENEINFO=NPPA:4878|NPPA-AS1:100379251; [...] CLNSIG=3;
CLNDSDB=MedGen:OMIM; CLNDSDBID=C2677294:612201; CLNDBN=Atrial_fibrillation\2c_familial\2c_6; [...]
13 32316475 rs80359298 CAA C . . RS=80359298; [...] GENEINFO=BRCA2:675; [...] CLNSIG=1|5; CLNDSDB=MedGen:
OMIM;SNOMED_CT|MedGen:OMIM; CLNDSDBID=C0346153:114480:254843006|C2675520:612555; CLNDBN=
Familial_cancer_of_breast|Breast-ovarian_cancer\2c_familial_2; [...]
```

Los campos de interés en cada línea son los siguientes:

- CHROM: Número de cromosoma.
- POS: Posición del SNP dentro del cromosoma (comienza a numerarse en 1).
- ID: Identificador único del SNP ('rsXXXX').
- REF: Base(s) que aparecen en esa posición en el genoma humano de referencia. En caso de que aparezca una pequeña cadena de varias bases (ejemplo: "ATTGGAG"), el SNP que se indica reemplaza esta secuencia de bases por una sola.
- ALT: la(s) base(s) alternativa(s) que se han observado en la población. Si se han observado distintas mutaciones para la misma posición, éstas se indican delimitadas por coma (ejemplo: "A,C,T").
- INFO: Este campo representa información adicional sobre el SNP en forma de listado de atributos separados por ';'. Entre estos atributos, destacamos por su interés los siguientes:
 - GENEINFO: Nombre e identificador del gen que contiene este SNP. Ejemplo: GENEINFO=ISG15:9636 (Nombre del gen: ISG15, Identificador del gen: 9636). En caso de que se trate de varios genes, se separan con '|' o ';'. Ejemplo: GENEINFO=B3GALT6:126792|SDF4:51150
 - CAF: Frecuencia con que se observa cada base descrita en este SNP en la población. Ejemplo: CAF=0.9912,0.008786 indica que la base de la referencia se observa con frecuencia 0.9912 y la base alternativa con frecuencia 0.008786. El primer valor de CAF corresponde a frecuencia de la base REF, los siguientes a las bases indicadas en ALT, en el mismo orden.
 - COMMON: Indica si es un SNP común en la población (0 - no, 1 - si).
 - CLNSIG: relevancia clínica del SNP: 0/1 - Incierta, Desconocida, 2 - Benigno, 3 - Probablemente benigno, 4 - Probablemente patógeno, 5 - Patógeno, 6 - Relevante en respuesta a fármaco, 7 - Histocompatibilidad, 255 - Otro. En caso de que el SNP esté asociado con varias enfermedades se mostrará un código CLNSIG para cada enfermedad (delimitados por '|' o ';'), o un solo código CLNSIG, indicando que la relevancia clínica del SNP es la misma para todas las enfermedades.

- CLNDBN: Nombre de la enfermedad asociada al SNP. También se suministran el ID único de la enfermedad (CLNDSDBID) y la base de datos que provee este ID (CLNDSDB). En caso de que un SNP esté asociado a varias enfermedades, éstas se separan con '|' o ';'. El siguiente ejemplo hace referencia a tres enfermedades: CLNDSDB=MedGen|MedGen:OMIM|MedGen; CLNDSDBID=CN178850|C3809288:615373|CN169374; CLNDBN=Dilated_cardiomyopathy_1LL|Left_ventricular_noncompaction_8|not_specified;

1.2. TDA enfermedad

Para relacionar SNPs con enfermedades proponemos la creación de una clase enfermedad, que deberá tener entre otros los métodos abajo indicados. La especificación de la clase enfermedad se realizará en el fichero [enfermedad.h](#) y la implementación de la clase enfermedad en el fichero [enfermedad.hxx](#).

```
class enfermedad {
private:
    string  name;          // nombre de la enfermedad. Almacenar completo en minúscula.
    string  ID;            // ID único para la enfermedad
    string  database;      // Base de datos que provee el ID

public:
    enfermedad (); //Constructor de enfermedad por defecto
    enfermedad (const string & name, const string & ID, const string & database);

    void setName(const string & name);
    void setID(const string & ID);
    void setDatabase(const string & database);

    string getName();
    string getID();
    string getDatabase();

    enfermedad & operator=(const enfermedad & e);
    string toString() const;

    // Operadores relacionales
    bool operator==(const enfermedad & e) const;
    bool operator!=(const enfermedad & e) const;
    bool operator<(const enfermedad & e) const; //Orden alfabético por campo name.

    bool nameContains(const string & str) const; //Devuelve True si str está incluido en el
        nombre de la enfermedad, aunque no se trate del nombre completo. No debe ser sensible a mayúsculas/minúsculas.
}

ostream& operator<< (ostream& os, const enfermedad & e); //imprime enfermedad

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

Así, podremos trabajar con enfermedades como indica el siguiente código

```
...
enfermedad e1("Breast-ovarian_cancer\x2c_familial_2", "C2675520:612555", "MedGen:OMIM");
enfermedad e2("Prostate_cancer\x2c_susceptibility_to", "", "");
enfermedad e3 = e1;
...
if (e1.nameContains("cancer"))
    cout << e1 << " es un tipo de cancer. ";
...
```

1.3. Mutación

A igual que con la clase enfermedad, la especificación del tipo mutación y su implementación se realizará en los ficheros [mutacion.h](#) y [mutacion.hxx](#), respectivamente, y debe tener la información de los atributos (con su representación asociada)

- chr: identificador del cromosoma (string). Los cromosomas válidos son: "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "X", "Y", "MT".
- pos: identificador de la posición dentro del cromosoma (unsigned int).
- ID: identificador del SNP/mutación (string).
- ref_alt: base(s) en el genoma de referencia y alternativa(s) posible(s) (vector de string). La primera posición la ocupará el string con la(s) base(s) del genoma de referencia, y, a continuación, aparecerán la(s) base(s) alternativas en el mismo orden que se indica en el fichero. Ejemplos:

```
X 154032548 rs61754422 C A,G,T
ref_alt: ["C", "A", "G", "T"]

1 1338032 rs797044840 GTAGGCAGG GC
ref_alt: ["GTAGGCAGG", "GC"]
```

- genes: gen(es) asociado(s) al SNP (vector de string). Ejemplo:

```
1 11847311 rs755212754 G A . . [...]GENEINFO=NPPA:4878|NPPA-AS1:100379251;[...]
genes: ["NPPA:4878", "NPPA-AS1:100379251"]
```

- common: indica si el SNP es común en la población (bool).
- caf: frecuencia de cada base del SNP en la población (vector de float). En primer lugar debe indicarse la frecuencia de la base 'ref' (posición 0 de ref-alt), seguida por las frecuencias de las bases alternativas indicadas en 'ref-alt', en el mismo orden. Ejemplo:

```
1 11847114 rs202102042 C T . . RS=202102042;[...];CAF=0.9998,0.0001997;COMMON=0
ref_alt: ["C", "T"]
caf: [0.9998, 0.0001997]
common: False
```

- enfermedades: enfermedades asociadas al SNP (vector de enfermedad).
- clnsig: relevancia clínica del SNP para cada enfermedad utilizando el código numérico del campo CLNSIG (vector de int). En caso de que existan varias enfermedades asociadas a la mutación, cada una de ellas puede presentar diferente código CLNSIG, por lo se deben almacenar en el vector clnsig en el mismo orden que las enfermedades asociadas. En caso de presentarse sólo un código CLNSIG y varias enfermedades, este código se aplica a todas ellas. Ejemplo:

```
13 32316475 rs80359298 CAA C . . RS=80359298;[...];CLNSIG=1|5;CLNDSDB=MedGen:OMIM:SNOMED_CT|MedGen:OMIM;
CLNDSDBID=C0346153:114480:254843006|C2675520:612555;CLNDBN=Familial_cancer_of_breast|Breast-
ovarian_cancer\x2c_familial_2;[...]

enfermedades: [ enfermedad("Familial_cancer_of_breast", "C0346153:114480:254843006", "
MedGen:OMIM:SNOMED_CT"),
                enfermedad("Breast-ovarian_cancer\x2c_familial_2", "C2675520:612555", "
MedGen:OMIM")]
clnsig: [1,5]
```

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

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

1.4. "Se Entrega / Se Pide"

1.4.1. 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.
- `documentacion.dox` Este fichero contiene el fichero de configuración de doxygen necesario para generar la documentación del proyecto (html y pdf). Para ello, basta con ejecutar desde la línea de comando

```
doxygen doxPractica.txt
```

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
```

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

- `mutacion.h` Plantilla para la especificación del TDA mutación
- `mutacion.hxx` Plantilla para la implementación del TDA mutación
- `enfermedad.h` Plantilla para la especificación del TDA enfermedad
- `enfermedad.hxx` Plantilla para la implementación del TDA enfermedad
- `principal.cpp` Fichero donde se incluye el main del programa. Este programa toma como entrada el fichero de datos `"clinvar_20160831.vcf"`, carga las mutaciones en un vector, muestra el número total de mutaciones leídas del fichero y el número de mutaciones que están asociadas a una enfermedad que indica el usuario.

1.4.2. Se Pide

- Diseñar la función de abstracción e invariante de la representación del tipo enfermedad.
- Diseñar la función de abstracción e invariante de la representación del tipo mutación.
- Implementar el código asociado a los ficheros `.hxx`.
- Implementar el código asociado a `principal.cpp`.

1.5. "Fecha Límite de Entrega"

La fecha límite de entrega será el 23 de Octubre a las 23:50 hrs.

2. Lista de tareas pendientes

Clase `enfermedad`

Implementa esta clase, junto con su documentación asociada

Clase `mutacion`

Implementa esta clase, junto con su documentación asociada

3. Índice de clases

3.1. Lista de clases

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

enfermedad	
Clase enfermedad, asociada al TDA enfermedad	8
mutacion	
Clase mutacion, asociada a la definición de una mutación/SNP	12

4. Índice de archivos

4.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

enfermedad.h	15
enfermedad.hxx	16
mutacion.h	17
principal.cpp	17

5. Documentación de las clases

5.1. Referencia de la Clase enfermedad

Clase enfermedad, asociada al TDA enfermedad.

```
#include <enfermedad.h>
```

Métodos públicos

- **enfermedad** ()
Fichero de implementacion de la clase enfermedad.
- **enfermedad** (const std::string &name, const std::string &ID, const std::string &database)
Constructor dados los parámetros.
- void **setName** (const std::string &name)
Cambia el nombre de la enfermedad.
- void **setID** (const std::string &ID)
Cambia el ID de la enfermedad.
- void **setDatabase** (const std::string &database)
Cambia el database de la enfermedad.
- std::string **getName** () const
Consultor name.
- std::string **getID** () const
Consultor ID.
- std::string **getDatabase** () const
Consultor database.
- **enfermedad** & **operator=** (const **enfermedad** &e)
Operador de asignación.

- `std::string toString () const`
Genera un string con los atributos de la enfermedad.
- `bool operator== (const enfermedad &e) const`
Compara los atributos de 'e' y la enfermedad que llama al método.
- `bool operator!= (const enfermedad &e) const`
Compara los atributos de 'e' y la enfermedad que llama al método.
- `bool operator< (const enfermedad &e) const`
Compara el nombre de 'e' y la enfermedad que llama al método.
- `bool nameContains (const std::string &str) const`
Indica si str está contenido en name.

Atributos privados

- `std::string name`
- `std::string ID`
- `std::string database`

5.1.1. Descripción detallada

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)

Tareas pendientes Implementa esta clase, junto con su documentación asociada

5.1.2. Documentación del constructor y destructor

5.1.2.1. `enfermedad::enfermedad ()`

Fichero de implementacion de la clase enfermedad.

Constructor por defecto, crea una enfermedad vacía

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

Constructor dados los parámetros.

Parámetros

<i>name</i>	Nombre enfermedad
<i>ID</i>	Identificador enfermedad
<i>database</i>	Base de datos que provee el ID

5.1.3. Documentación de las funciones miembro

5.1.3.1. `std::string enfermedad::getDatabase () const`

Consultor database.

Devuelve

Database de la enfermedad

5.1.3.2. `std::string enfermedad::getID () const`

Consultor ID.

Devuelve

ID de la enfermedad

5.1.3.3. `std::string enfermedad::getName () const`

Consultor name.

Devuelve

Nombre de la enfermedad

5.1.3.4. `bool enfermedad::nameContains (const std::string & str) const`

Indica si str está contenido en name.

Parámetros

<i>str</i>	String que buscamos
------------	---------------------

Devuelve

Booleano indicando si son iguales

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

Compara los atributos de 'e' y la enfermedad que llama al método.

Parámetros

<i>e</i>	Enfermedad con la que comparar
----------	--------------------------------

Devuelve

Booleano indicando si son distintos

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

Compara el nombre de 'e' y la enfermedad que llama al método.

Parámetros

<i>e</i>	Enfermedad con la que comparar
----------	--------------------------------

Devuelve

Booleano indicando orden lexicográfico de los name

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

Operador de asignación.

Parámetros

<i>e</i>	Enfermedad que se asigna
----------	--------------------------

Devuelve

Referencia a enfermedad que llama al método

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

Compara los atributos de 'e' y la enfermedad que llama al método.

Parámetros

<i>e</i>	Enfermedad con la que comparar
----------	--------------------------------

Devuelve

Booleano indicando si son iguales

5.1.3.9. `void enfermedad::setDatabase (const std::string & database)`

Cambia el database de la enfermedad.

Parámetros

<i>database</i>	Nuevo database
-----------------	----------------

5.1.3.10. `void enfermedad::setID (const std::string & ID)`

Cambia el ID de la enfermedad.

Parámetros

<i>ID</i>	Nuevo ID
-----------	----------

5.1.3.11. void enfermedad::setName (const std::string & *name*)

Cambia el nombre de la enfermedad.

Parámetros

<i>name</i>	Nuevo nombre
-------------	--------------

5.1.3.12. std::string enfermedad::toString () const

Genera un string con los atributos de la enfermedad.

Devuelve

String con los atributos

5.1.4. Documentación de los datos miembro**5.1.4.1. std::string enfermedad::database [private]****5.1.4.2. std::string enfermedad::ID [private]****5.1.4.3. std::string enfermedad::name [private]**

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

- [enfermedad.h](#)
- [enfermedad.hxx](#)

5.2. Referencia de la Clase mutacion

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

```
#include <mutacion.h>
```

Métodos públicos

- `mutacion ()`
- `mutacion (const mutacion &m)`
- `mutacion (const std::string &str)`
- `void setID (const std::string &id)`
- `void setChr (const std::string &chr)`
- `void setPos (const unsigned int &pos)`
- `void setRef_alt (const std::vector< std::string > &ref_alt)`
- `void setGenes (const std::vector< std::string > &genes)`
- `void setCommon (const bool &common)`
- `void setCaf (const std::vector< float > &caf)`
- `void setEnfermedades (const std::vector< enfermedad > &enfermedades)`
- `void setClnsig (const std::vector< int > &clnsig)`
- `std::string getID () const`
- `std::string getChr () const`
- `unsigned int getPos () const`
- `const std::vector< std::string > & getRef_alt () const`
- `const std::vector< std::string > & getGenes () const`
- `bool getCommon () const`
- `const std::vector< float > & getCaf () const`
- `const std::vector< enfermedad > & getEnfermedades () const`
- `const std::vector< int > & getClnsig () const`
- `mutacion & operator= (const mutacion &m)`
- `bool operator== (const mutacion &m) const`
- `bool operator< (const mutacion &m) const`

Métodos privados

- `void copiar (const mutacion &m)`
- `int numCromosoma () const`

Atributos privados

- `std::string ID`
- `std::string chr`
- `unsigned int pos`
- `std::vector< std::string > ref_alt`
- `std::vector< std::string > genes`
- `bool common`
- `std::vector< float > caf`
- `std::vector< enfermedad > enfermedades`
- `std::vector< int > clnsig`

5.2.1. Descripción detallada

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

`mutacion::mutacion,`

Tareas pendientes Implementa esta clase, junto con su documentación asociada

5.2.2. Documentación del constructor y destructor

5.2.2.1. `mutacion::mutacion ()`

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

5.2.2.3. `mutacion::mutacion (const std::string & str)`

5.2.3. Documentación de las funciones miembro

5.2.3.1. `void mutacion::copiar (const mutacion & m) [private]`

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

5.2.3.3. `std::string mutacion::getChr () const`

5.2.3.4. `const std::vector<int>& mutacion::getCInsig () const`

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

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

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

5.2.3.8. `std::string mutacion::getID () const`

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

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

5.2.3.11. `int mutacion::numCromosoma () const [private]`

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

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

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

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

5.2.3.16. `void mutacion::setChr (const std::string & chr)`

5.2.3.17. `void mutacion::setCInsig (const std::vector< int > & cInsig)`

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

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

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

5.2.3.21. `void mutacion::setID (const std::string & id)`

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

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

5.2.4. Documentación de los datos miembro

5.2.4.1. `std::vector<float> mutacion::caf [private]`

5.2.4.2. `std::string mutacion::chr [private]`

5.2.4.3. `std::vector<int> mutacion::cInsig [private]`

5.2.4.4. `bool mutacion::common [private]`

5.2.4.5. `std::vector<enfermedad> mutacion::enfermedades [private]`

5.2.4.6. `std::vector<std::string> mutacion::genes [private]`

5.2.4.7. `std::string mutacion::ID [private]`

5.2.4.8. `unsigned int mutacion::pos [private]`

5.2.4.9. `std::vector<std::string> mutacion::ref_alt [private]`

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

- [mutacion.h](#)

6. Documentación de archivos

6.1. Referencia del Archivo documentacion.dox

6.2. Referencia del Archivo enfermedad.h

```
#include <string>
#include <iostream>
#include "enfermedad.hxx"
```

Clases

- class [enfermedad](#)
Clase enfermedad, asociada al TDA enfermedad.

Funciones

- `std::ostream & operator<< (std::ostream &os, const enfermedad &e)`
Manda los datos de la enfermedad 'e' por os.
- `std::string aMinuscula (std::string str)`
Transforma un string a minúscula.

6.2.1. Documentación de las funciones

6.2.1.1. `std::string aMinuscula (std::string str)`

Transforma un string a minúscula.

Parámetros

<i>str</i>	String que cambiamos
------------	----------------------

Devuelve

El string con las letras en minúscula

6.2.1.2. `std::ostream& operator<< (std::ostream & os, const enfermedad & e)`

Manda los datos de la enfermedad 'e' por os.

Parámetros

<i>os</i>	Flujo de salida que utilizamos
<i>e</i>	Enfermedad de la que se envían los datos

Devuelve

Flujo de salida os por referencia

6.3. Referencia del Archivo enfermedad.hxx

Funciones

- `std::ostream & operator<< (std::ostream &os, const enfermedad &e)`

Manda los datos de la enfermedad 'e' por os.

- `std::string aMinuscula (std::string str)`

Transforma un string a minúscula.

6.3.1. Documentación de las funciones

6.3.1.1. `std::string aMinuscula (std::string str)`

Transforma un string a minúscula.

Parámetros

<i>str</i>	String que cambiamos
------------	----------------------

Devuelve

El string con las letras en minúscula

6.3.1.2. `std::ostream& operator<< (std::ostream & os, const enfermedad & e)`

Manda los datos de la enfermedad 'e' por os.

Parámetros

<i>os</i>	Flujo de salida que utilizamos
<i>e</i>	Enfermedad de la que se envían los datos

Devuelve

Flujo de salida *os* por referencia

6.4. Referencia del Archivo mutacion.h

```
#include <string>
#include <iostream>
#include <vector>
#include "enfermedad.h"
#include "mutacion.hxx"
```

Clases

- class `mutacion`

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

Funciones

- `std::ostream & operator<< (std::ostream &, const mutacion &)`

6.4.1. Documentación de las funciones

6.4.1.1. `std::ostream& operator<< (std::ostream & , const mutacion &)`

6.5. Referencia del Archivo principal.cpp

```
#include "mutacion.h"
#include "enfermedad.h"
#include <iostream>
#include <fstream>
#include <vector>
```

Funciones

- bool `load` (vector< `mutacion` > &vm, const string &s)
lee un fichero de mutaciones, línea a línea
- int `cuentaMutacionesEnfermedad` (vector< `mutacion` > &vm, const string &s)
Recorre un vector de mutaciones y devuelve cuántas de estas mutaciones están asociadas a un nombre de enfermedad s.
- int `main` (int argc, char *argv[])

6.5.1. Documentación de las funciones

6.5.1.1. `int cuentaMutacionesEnfermedad (vector< mutacion > &vm, const string &s)`

Recorre un vector de mutaciones y devuelve cuántas de estas mutaciones están asociadas a un nombre de enfermedad s.

Parámetros

<i>in</i>	<i>vm</i>	vector de mutaciones
<i>in</i>	<i>s</i>	texto asociado al nombre de la enfermedad.

Devuelve

int número de mutaciones asociadas a enfermedades cuyo nombre contiene *s*

6.5.1.2. `bool load (vector< mutacion > & vm, const string & s)`

lee un fichero de mutaciones, linea a linea

Parámetros

<i>in</i>	<i>s</i>	nombre del fichero
<i>in, out</i>	<i>vm</i>	vector sobre el que se lee

Devuelve

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

6.5.1.3. `int main (int argc, char * argv[])`

Índice alfabético

aMinuscula
 enfermedad.h, 15
 enfermedad.hxx, 16

caf
 mutacion, 15

chr
 mutacion, 15

clnsig
 mutacion, 15

common
 mutacion, 15

copiar
 mutacion, 14

cuentaMutacionesEnfermedad
 principal.cpp, 17

database
 enfermedad, 12

documentacion.dox, 15

enfermedad, 8
 database, 12
 enfermedad, 9
 getDatabase, 9
 getID, 10
 getName, 10
 ID, 12
 name, 12
 nameContains, 10
 operator!=, 10
 operator<, 10
 operator=, 11
 operator==, 11
 setDatabase, 11
 setID, 11
 setName, 12
 toString, 12

enfermedad.h, 15
 aMinuscula, 15
 operator<<, 16

enfermedad.hxx, 16
 aMinuscula, 16
 operator<<, 16

enfermedades
 mutacion, 15

genes
 mutacion, 15

getCaf
 mutacion, 14

getChr
 mutacion, 14

getClnsig
 mutacion, 14

getCommon
 mutacion, 14

getDatabase
 enfermedad, 9

getEnfermedades
 mutacion, 14

getGenes
 mutacion, 14

getID
 enfermedad, 10
 mutacion, 14

getName
 enfermedad, 10

getPos
 mutacion, 14

getRef_alt
 mutacion, 14

ID
 enfermedad, 12
 mutacion, 15

load
 principal.cpp, 18

main
 principal.cpp, 18

mutacion, 12
 caf, 15
 chr, 15
 clnsig, 15
 common, 15
 copiar, 14
 enfermedades, 15
 genes, 15
 getCaf, 14
 getChr, 14
 getClnsig, 14
 getCommon, 14
 getEnfermedades, 14
 getGenes, 14
 getID, 14
 getPos, 14
 getRef_alt, 14
 ID, 15
 mutacion, 14
 numCromosoma, 14
 operator<, 14
 operator=, 14
 operator==, 14
 pos, 15
 ref_alt, 15
 setCaf, 14
 setChr, 14
 setClnsig, 14
 setCommon, 14
 setEnfermedades, 14

- setGenes, [14](#)
- setID, [14](#)
- setPos, [15](#)
- setRef_alt, [15](#)
- mutacion.h, [17](#)
- operator<<, [17](#)
- name
 - enfermedad, [12](#)
- nameContains
 - enfermedad, [10](#)
- numCromosoma
 - mutacion, [14](#)
- operator!=
 - enfermedad, [10](#)
- operator<
 - enfermedad, [10](#)
 - mutacion, [14](#)
- operator<<
 - enfermedad.h, [16](#)
 - enfermedad.hxx, [16](#)
 - mutacion.h, [17](#)
- operator=
 - enfermedad, [11](#)
 - mutacion, [14](#)
- operator==
 - enfermedad, [11](#)
 - mutacion, [14](#)
- pos
 - mutacion, [15](#)
- principal.cpp, [17](#)
 - cuentaMutacionesEnfermedad, [17](#)
 - load, [18](#)
 - main, [18](#)
- ref_alt
 - mutacion, [15](#)
- setCaf
 - mutacion, [14](#)
- setChr
 - mutacion, [14](#)
- setClnsig
 - mutacion, [14](#)
- setCommon
 - mutacion, [14](#)
- setDatabase
 - enfermedad, [11](#)
- setEnfermedades
 - mutacion, [14](#)
- setGenes
 - mutacion, [14](#)
- setID
 - enfermedad, [11](#)
 - mutacion, [14](#)
- setName
 - enfermedad, [12](#)
- setPos
 - mutacion, [15](#)
- setRef_alt
 - mutacion, [15](#)
- toString
 - enfermedad, [12](#)