TDA Mutación y TDA Enfermedad

V1

Generado por Doxygen 1.8.11

Índice

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 deleción 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 (fichero: 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\x2c_familial\x2c_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\x2c_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, estás 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; CL↩ NDSDBID=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 representacion 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. Índice de clases

2.1. Lista de clases

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

3. Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

4. Documentación de las clases

4.1. Referencia de la Clase enfermedad

Clase enfermedad, asociada al TDA enfermedad.

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

Métodos públicos

- enfermedad ()

  variable que contine la base de datos correspondiente a la enfermedad
- 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 () const
- string getID () const
- string getDatabase () const
- enfermedad & operator= (const enfermedad &e)
- string toString () const
- bool operator== (const enfermedad &e) const
- bool operator!= (const enfermedad &e) const
- bool operator< (const enfermedad &e) const
- bool nameContains (const string &str) const

Atributos privados

- string name
- string ID

  variable que contiene el nombre de la enfermedad
- string database

  variable que contiene el ID correspondiente a la enfermedad

4.1.1. 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)

4.1.2. Documentación del constructor y destructor

4.1.2.1. enfermedad::enfermedad ( )

variable que contine la base de datos correspondiente a la enfermedad

Constructor por defecto de la clase enfermedad

4.1.2.2. enfermedad::enfermedad (const string & *name,* const string & *ID,* const string & *database*)

Constructor con parámetros, recibe todos los campos necesarios y inicializa con sus respectivos valores

4.1.3. Documentación de las funciones miembro

4.1.3.1. string enfermedad::getDatabase ( ) const

Método consultor de la variable privada database, devuelve la mismo.

Devuelve

Variable privada database

4.1.3.2. string enfermedad::getID ( ) const

Método consultor de la variable privada ID, devuelve el mismo.

Devuelve

Variable privada ID

4.1.3.3. string enfermedad::getName ( ) const

Método consultor de la variable privada name, devuelve el mismo.

Devuelve

Variable privada name

4.1.3.4. bool enfermedad::nameContains (const string & *str*) const

método para comprobar si nuestra enfermedad contiene en la variable de tipo string una subcadena pasada por parámetro

Parámetros

in	*String*	

Devuelve

En caso de contener la subcadena devuelve true

4.1.3.5. bool enfermedad::operator!= (const **enfermedad** & *e*) const

Operador sobrecargado !=, totalmente opuesto al anterior, compara el campo ID

Parámetros

in	*e*	objeto enfermedad

Devuelve

En caso de ser distintas enfermedades, devuelve true

4.1.3.6. bool enfermedad::operator< (const **enfermedad** & *e*) const

Operador sobrecargado $<$, en este caso comparamos dos string correspondientes al campo name de cada enfermedad y nos dice si están almacenados por orden alfabético

Parámetros

in	*e*	objeto enfermedad

Devuelve

En caso de estar odenadas devuelve true en otro caso false

4.1.3.7. **enfermedad** & enfermedad::operator= (const **enfermedad** & *e*)

Operador sobrecargado =, este operador recibe como parámetro una enfermedad e iguala los campos de la primera enfermedad a los parámetros de la segunda enfermedad —$>$ e1 = e2 e1 es la primera enfermedad y a la que se le asocian los campos de la segunda enfermedad la cual es e2

Parámetros

in	*e*	objeto enfermedad

Devuelve

Objeto enfermedad inicial ya igualado al pasado por parámetro

4.1.3.8. bool enfermedad::operator== (const enfermedad & *e*) const

Operador sobrecargado ==, este operador recibe como parámetro una enfermedad y comprueba que el campo ID de la misma sea el mismo que el ID de la enfermedad 1, es decir, comprueba si son la misma enfermedad.

Parámetros

in	*e*	objeto enfermedad

Devuelve

En caso de ser iguales las enfermedades, devuelve true

4.1.3.9. void enfermedad::setDatabase (const string & *database*)

Método permitido para la edición y inicialización de la variable privada database de la clase enfermedad.

Parámetros

in	*string*	database

4.1.3.10. void enfermedad::setID (const string & *ID*)

Método permitido para la edición y inicialización de la variable privada ID de la clase enfermedad.

Parámetros

in	*string*	ID

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

Método permitido para la edición y inicialización de la variable privada name de la clase enfermedad.

Parámetros

in	*string*	name

4.1.3.12. string enfermedad::toString ( ) const

Método para mostrar en pantalla la informacion del objeto asociado a dicho método

Devuelve

String con name , ID, database

4.1.4. Documentación de los datos miembro

4.1.4.1. string enfermedad::database [private]

variable que contiene el ID correspondiente a la enfermedad

4.1.4.2. string enfermedad::ID [private]

variable que contiene el nombre de la enfermedad

4.1.4.3. string enfermedad::name [private]

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

- enfermedad.h
- enfermedad.hxx

4.2. Referencia de la Clase mutacion

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

`#include <mutacion.h>`

Métodos públicos

- mutacion ()
  variable que contiene los numerosos clnsig de cada mutación
- mutacion (const mutacion &m)
- mutacion (const string &str)
- void setID (const string &id)
- void setChr (const string &chr)
- void setPos (const unsigned int &pos)
- void setRef_alt (const std::vector< string > &ref_alt)
- void setGenes (const std::vector< 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)
- string getID () const
- string getChr () const
- unsigned int getPos () const
- const std::vector< string > & getRef_alt () const
- const std::vector< 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

Atributos privados

- string ID
- string chr

  variable que contiene el ID de la mutacion
- unsigned int pos

  variable que contiene el cromosoma concreto
- std::vector< string > ref_alt

  variable que contine la posición de la mutación
- std::vector< string > genes

  variable que contiene los numerosos ref_caf de cada mutación
- bool common

  variable que contiene los numerosos genes de cada mutación
- std::vector< float > caf

  variable que nos dice si es común o no
- std::vector< enfermedad > enfermedades

  variable que contiene los numerosos caf de cada mutación
- std::vector< int > clnsig

  variable que contiene las numerosas enfermedades de cada mutación

4.2.1. Descripción detallada

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

mutacion::mutacion

4.2.2. Documentación del constructor y destructor

4.2.2.1. mutacion::mutacion ( )

variable que contiene los numerosos clnsig de cada mutación

Constructor por defecto de la clase mutación

4.2.2.2. mutacion::mutacion (const mutacion & *m*)

Constructor copia, recibe recibe una mutación y crea una copia de esta

4.2.2.3. mutacion::mutacion (const string & *str*)

Constructor a traves de un string, recibe recibe una cadena con toda la información y crea un objeto a raíz de esta

Parámetros

in	*string*	str

4.2.3. Documentación de las funciones miembro

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

Método consultor de la variable privada caf, devuelve los mismos.

Devuelve

Variable privada caf

4.2.3.2. string mutacion::getChr ( ) const

Método consultor de la variable privada chr, devuelve el mismo.

Devuelve

Variable privada chr

4.2.3.3. const std::vector< int > & mutacion::getClnsig ( ) const

Método consultor de la variable privada clnsig, devuelve el mismo.

Devuelve

Variable privada clnsig

4.2.3.4. bool mutacion::getCommon ( ) const

Método consultor de la variable privada common, devuelve el mismo.

Devuelve

Variable privada common

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

Método consultor de la variable privada enfermedades, devuelve las mismas.

Devuelve

Variable privada enfermedades

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

Método consultor de la variable privada genes, devuelve los mismos.

Devuelve

Variable privada genes

4.2.3.7. string mutacion::getID ( ) const

Método consultor de la variable privada ID, devuelve el mismo.

Devuelve

Variable privada ID

4.2.3.8. unsigned int mutacion::getPos ( ) const

Método consultor de la variable privada pos, devuelve la misma.

Devuelve

Variable privada pos

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

Método consultor de la variable privada ref_alt, devuelve el mismo.

Devuelve

Variable privada ref_alt

4.2.3.10. bool mutacion::operator< (const mutacion & *m*) const

Operador sobrecargado <, en este caso comparamos dos string correspondientes al campo chr de cada mutacion y nos dice si están almacenados por orden, según el criterio de cromososa y según el criterio de pos, este segundo en caso de que el primero no nos aclare las dudas

Parámetros

in	*m*	objeto mutación

Devuelve

true en caso de estar ordenadas, false en el contrario

4.2.3.11. mutacion & mutacion::operator= (const mutacion & *m*)

Operador sobrecargado =, este operador recibe como parámetro una mutación m iguala los campos de la primera mutación a los parámetros de la segunda mutación —> m1 = m2 m1 es la primera mutación y a la que se le asocian los campos de la segunda mutación la cual es m2

Parámetros

in	*m*	objeto mutación

Devuelve

Objeto mutación inicial ya igualado al pasado por parámetro

4.2.3.12. bool mutacion::operator== (const mutacion & *m*) const

Operador sobrecargado ==, este operador recibe como parámetro una mutacion y comprueba que el campo ID de la misma sea el mismo que el ID de la mutacion 1, es decir, comprueba si son la misma mutacion

Parámetros

in	*m*	objeto mutación

Devuelve

true en caso de ser la misma enfermedad, false en el contrario

4.2.3.13. void mutacion::setCaf (const std::vector< float > & *caf*)

Método permitido para la edición y inicialización de la variable privada caf de la clase mutación.

Parámetros

in	*vector<float>*	caf

4.2.3.14. void mutacion::setChr (const string & *chr*)

Método permitido para la edición y inicialización de la variable privada chr de la clase mutacion.

Parámetros

in	*string*	chr

4.2.3.15. void mutacion::setClnsig (const std::vector< int > & *clnsig*)

Método permitido para la edición y inicialización de la variable privada clnsig de la clase mutación.

Parámetros

in	*vector<int>*	clnsig

4.2.3.16. void mutacion::setCommon (const bool & *common*)

Método permitido para la edición y inicialización de la variable privada common de la clase mutación.

Parámetros

in	*bool*	common

4.2.3.17. void mutacion::setEnfermedades (const std::vector< enfermedad > & *enfermedades*)

Método permitido para la edición y inicialización de la variable privada enfermedades de la clase mutación.

Parámetros

in	*vector<enfermedad>*	enfermedades

4.2.3.18. void mutacion::setGenes (const std::vector< string > & *genes*)

Método permitido para la edición y inicialización de la variable privada genes de la clase mutación.

Parámetros

in	*vector<string>*	genes

4.2.3.19. void mutacion::setID (const string & *id*)

Método permitido para la edición y inicialización de la variable privada ID de la clase mutación.

Parámetros

in	*string*	ID

4.2.3.20. void mutacion::setPos (const unsigned int & *pos*)

Método permitido para la edición y inicialización de la variable privada pos de la clase mutacion.

Parámetros

in	*unsigned*	int pos

4.2.3.21. void mutacion::setRef_alt (const std::vector< string > & *ref_alt*)

Método permitido para la edición y inicialización de la variable privada ref_alt de la clase mutación.

Parámetros

in	*vector<string>*	ref_alt

4.2.4. Documentación de los datos miembro

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

variable que nos dice si es común o no

4.2.4.2. string mutacion::chr [private]

variable que contiene el ID de la mutacion

4.2.4.3. std::vector<int> mutacion::clnsig [private]

variable que contiene las numerosas enfermedades de cada mutación

4.2.4.4. bool mutacion::common [private]

variable que contiene los numerosos genes de cada mutación

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

variable que contiene los numerosos caf de cada mutación

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

variable que contiene los numerosos ref_caf de cada mutación

4.2.4.7. string mutacion::ID [private]

4.2.4.8. unsigned int mutacion::pos [private]

variable que contiene el cromosoma concreto

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

variable que contine la posición de la mutación

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

- mutacion.h
- mutacion.hxx

5. Documentación de archivos

5.1. Referencia del Archivo documentacion.dox

5.2. Referencia del Archivo enfermedad.h

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

Clases

- class enfermedad

  Clase enfermedad, asociada al TDA enfermedad.

Funciones

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

5.2.1. Documentación de las funciones

5.2.1.1. ostream& operator<< (ostream & *os,* const enfermedad & *e*)

método externo a la clase el cual nos sirve para mostrar toda la informacion de la enfermedad correspondiente, funciona en consonancia con el método toString()

Parámetros

in	*e*	objeto enfermedad

Devuelve

Devuelve un flujo de salida con los datos de la enfermedad

5.3. Referencia del Archivo enfermedad.hxx

Funciones

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

5.3.1. Documentación de las funciones

5.3.1.1. ostream& operator<< (ostream & *os,* const enfermedad & *e*)

método externo a la clase el cual nos sirve para mostrar toda la informacion de la enfermedad correspondiente, funciona en consonancia con el método toString()

Parámetros

in	*e*	objeto enfermedad

Devuelve

Devuelve un flujo de salida con los datos de la enfermedad

5.4. Referencia del Archivo mutacion.h

```
#include <sstream>
#include <vector>
#include <cstdlib>
#include "enfermedad.h"
#include "mutacion.hxx"
```

Clases

- class mutacion

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

Funciones

- ostream & operator<< (ostream &, const mutacion &)

5.4.1. Documentación de las funciones

5.4.1.1. ostream& operator<< (ostream & , const mutacion &)

método externo a la clase el cual nos sirve para mostrar toda la informacion de la mutación correspondiente

Parámetros

in	*m*	objeto mutacion

Devuelve

Devuelve toda la información desglosada del objeto pasado por parámetro

5.5. Referencia del Archivo mutacion.hxx

Funciones

- ostream & operator<< (ostream &os, const mutacion &mutacion)

5.5.1. Documentación de las funciones

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

método externo a la clase el cual nos sirve para mostrar toda la informacion de la mutación correspondiente

Parámetros

in	*m*	objeto mutacion

Devuelve

Devuelve toda la información desglosada del objeto pasado por parámetro

5.6. 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, linea a linea

- 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[])

5.6.1. Documentación de las funciones

5.6.1.1. 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

in	*vm*	vector de mutaciones
in	*s*	texto asociado al nombre de la enfermedad.

Devuelve

int número de mutaciones asociadas a enfermedades cuyo nombre contiene s

5.6.1.2. bool load (vector< mutacion > & *vm*, const string & *s*)

lee un fichero de mutaciones, linea a linea

Parámetros

in	*s*	nombre del fichero
in,out	*vm*	vector sobre el que se lee

Devuelve

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

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

Índice alfabético