TDA Juego de las Letras

Generated by Doxygen 1.9.2

1 Class Index 1

1.1 Class List . . . 1

2 File Index 3

2.1 File List . . . 3

3 Class Documentation 5

3.1 bolsaLetras Class Reference . . . 5

3.1.1 Detailed Description . . . 6

3.1.2 Constructor & Destructor Documentation . . . 6

3.1.2.1 bolsaLetras() [1/2] . . . 6

3.1.2.2 bolsaLetras() [2/2] . . . 7

3.1.3 Member Function Documentation . . . 7

3.1.3.1 aniadeLetra() [1/2] . . . 7

3.1.3.2 aniadeLetra() [2/2] . . . 8

3.1.3.3 begin() [1/2] . . . 8

3.1.3.4 begin() [2/2] . . . 9

3.1.3.5 end() [1/2] . . . 9

3.1.3.6 end() [2/2] . . . 9

3.1.3.7 Esta() [1/2] . . . 9

3.1.3.8 Esta() [2/2] . . . 10

3.1.3.9 getBolsa() . . . 10

3.1.3.10 getLetras() . . . 11

3.1.3.11 getSoluciones() . . . 11

3.1.3.12 palabraValida() . . . 12

3.1.3.13 quitaLetra() [1/2] . . . 13

3.1.3.14 quitaLetra() [2/2] . . . 13

3.1.3.15 tama() . . . 14

3.1.4 Friends And Related Function Documentation . . . 14

3.1.4.1 operator<< . . . 14

3.1.4.2 operator>> . . . 15

3.2 conjuntoLetras Class Reference . . . 15

3.2.1 Detailed Description . . . 16

3.2.2 Member Function Documentation . . . 17

3.2.2.1 aniadeLetra() . . . 17

3.2.2.2 begin() [1/2] . . . 17

3.2.2.3 begin() [2/2] . . . 17

3.2.2.4 contarLetras() . . . 18

3.2.2.5 end() [1/2] . . . 19

3.2.2.6 end() [2/2] . . . 19

3.2.2.7 Esta() . . . 19

3.2.2.8 frecuenciaTotal() . . . 20

3.2.2.9 getLetra() [1/2] . . . 20

3.2.2.10 getLetra() [2/2] . . . 21
3.2.2.11 getLetras() . . . 21
3.2.2.12 PuntuacionPalabra() . . . 21
3.2.2.13 quitaLetra() . . . 22
3.2.2.14 tama() . . . 23
3.2.3 Friends And Related Function Documentation . . . 23
3.2.3.1 operator<< . . . 23
3.2.3.2 operator>> . . . 24
3.2.4 Member Data Documentation . . . 24
3.2.4.1 formato . . . 24
3.3 Diccionario Class Reference . . . 24
3.3.1 Detailed Description . . . 25
3.3.2 Constructor & Destructor Documentation . . . 25
3.3.2.1 Diccionario() . . . 25
3.3.3 Member Function Documentation . . . 25
3.3.3.1 aniade() . . . 25
3.3.3.2 begin() [1/2] . . . 26
3.3.3.3 begin() [2/2] . . . 26
3.3.3.4 end() [1/2] . . . 26
3.3.3.5 end() [2/2] . . . 26
3.3.3.6 Esta() . . . 26
3.3.3.7 operator=() . . . 27
3.3.3.8 PalabrasLongitud() . . . 27
3.3.3.9 size() . . . 28
3.3.4 Friends And Related Function Documentation . . . 28
3.3.4.1 operator<< . . . 28
3.3.4.2 operator>> [1/2] . . . 28
3.3.4.3 operator>> [2/2] . . . 29
3.4 bolsaLetras::iterator Class Reference . . . 29
3.4.1 Detailed Description . . . 29
3.4.2 Member Function Documentation . . . 29
3.4.2.1 operator"!=() . . . 29
3.4.2.2 operator*() . . . 30
3.4.2.3 operator++() . . . 30
3.4.2.4 operator==() . . . 30
3.4.3 Friends And Related Function Documentation . . . 31
3.4.3.1 bolsaLetras . . . 31
3.5 conjuntoLetras::iterator Class Reference . . . 31
3.5.1 Detailed Description . . . 32
3.5.2 Member Function Documentation . . . 32
3.5.2.1 operator"!=() . . . 32
3.5.2.2 operator*() . . . 32

3.5.2.3 operator++() . . . 33
3.5.2.4 operator==() . . . 33
3.5.3 Friends And Related Function Documentation . . . 33
3.5.3.1 conjuntoLetras . . . 33
3.6 Diccionario::iterator Class Reference . . . 34
3.6.1 Detailed Description . . . 34
3.6.2 Member Function Documentation . . . 34
3.6.2.1 operator"!=() . . . 34
3.6.2.2 operator*() . . . 34
3.6.2.3 operator++() . . . 35
3.6.2.4 operator==() . . . 35
3.6.3 Friends And Related Function Documentation . . . 35
3.6.3.1 Diccionario . . . 35
3.7 Letra Class Reference . . . 35
3.7.1 Detailed Description . . . 36
3.7.2 Constructor & Destructor Documentation . . . 36
3.7.2.1 Letra() [1/2] . . . 36
3.7.2.2 Letra() [2/2] . . . 36
3.7.3 Member Function Documentation . . . 37
3.7.3.1 getCaracter() . . . 37
3.7.3.2 getFrecuencia() . . . 37
3.7.3.3 getPuntuacion() . . . 37
3.7.3.4 operator<() . . . 37
3.7.3.5 operator==() . . . 38
3.7.3.6 setCaracter() . . . 39
3.7.3.7 setFrecuencia() . . . 39
3.7.3.8 setPuntuacion() . . . 40
3.7.4 Friends And Related Function Documentation . . . 40
3.7.4.1 operator<< . . . 40
3.7.4.2 operator>> . . . 40

4 File Documentation **43**

4.1 include/bolsaLetras.h File Reference . . . 43
4.1.1 Detailed Description . . . 44
4.2 bolsaLetras.h . . . 44
4.3 include/conjuntoLetras.h File Reference . . . 45
4.3.1 Detailed Description . . . 46
4.4 conjuntoLetras.h . . . 46
4.5 include/diccionario.h File Reference . . . 47
4.5.1 Detailed Description . . . 48
4.5.2 Function Documentation . . . 49
4.5.2.1 intercalado() . . . 49

4.6 diccionario.h . . . 49
4.7 include/letra.h File Reference . . . 50
4.7.1 Detailed Description . . . 51
4.8 letra.h . . . 52
4.9 src/bolsaLetras.cpp File Reference . . . 52
4.9.1 Detailed Description . . . 53
4.9.2 Function Documentation . . . 53
4.9.2.1 operator<<() . . . 53
4.9.2.2 operator>>() . . . 54
4.10 bolsaLetras.cpp . . . 54
4.11 src/cantidadLetras.cpp File Reference . . . 56
4.11.1 Detailed Description . . . 57
4.11.2 Function Documentation . . . 57
4.11.2.1 main() . . . 58
4.12 cantidadLetras.cpp . . . 59
4.13 src/conjuntoLetras.cpp File Reference . . . 61
4.13.1 Detailed Description . . . 61
4.13.2 Function Documentation . . . 61
4.13.2.1 operator<<() . . . 61
4.13.2.2 operator>>() . . . 62
4.14 conjuntoLetras.cpp . . . 63
4.15 src/diccionario.cpp File Reference . . . 65
4.15.1 Detailed Description . . . 65
4.15.2 Function Documentation . . . 65
4.15.2.1 intercalado() . . . 65
4.15.2.2 operator<<() . . . 66
4.15.2.3 operator>>() [1/2] . . . 66
4.15.2.4 operator>>() [2/2] . . . 66
4.16 diccionario.cpp . . . 67
4.17 src/letra.cpp File Reference . . . 68
4.17.1 Detailed Description . . . 68
4.17.2 Function Documentation . . . 68
4.17.2.1 operator<<() . . . 68
4.17.2.2 operator>>() . . . 69
4.18 letra.cpp . . . 69
4.19 src/testDiccionario.cpp File Reference . . . 70
4.19.1 Detailed Description . . . 71
4.19.2 Function Documentation . . . 71
4.19.2.1 main() . . . 72
4.20 testDiccionario.cpp . . . 73
4.21 src/testLetras.cpp File Reference . . . 74
4.21.1 Detailed Description . . . 74

4.21.2 Function Documentation 75
4.21.2.1 main() 75
4.22 testLetras.cpp 76

Index **79**

Chapter 1

Class Index

1.1 Class List

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

bolsaLetras 5
conjuntoLetras 15
Diccionario 24
bolsaLetras::iterator 29
conjuntoLetras::iterator 31
Diccionario::iterator 34
Letra 35

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

include/bolsaLetras.h
T.D.A. bolsaLetras . . . 43
include/conjuntoLetras.h
T.D.A Clase ConjuntoLetras que gestiona los archivos letra.txt . . . 45
include/diccionario.h
T.D.A Clase Diccionario . . . 47
include/letra.h
T.D.A. Clase Letra Clase letra que representa la unidad mínima del juego. Formada por: Caracter Frecuencia Puntuacion . . . 50
src/bolsaLetras.cpp . . . 52
src/cantidadLetras.cpp
Programa de prueba donde a partir de un diccionario y un conjunto de letras, genera un archivo de salida con las frecuencias absolutas y relativas de cada letra del diccionario. Ademas actualiza las cantidades y puntuaciones del archivo conjunto de letras(letras.txt) . . . 56
src/conjuntoLetras.cpp . . . 61
src/diccionario.cpp . . . 65
src/letra.cpp . . . 68
src/testDiccionario.cpp
Archivo de prueba para el diccionario . . . 70
src/testLetras.cpp
Archivo de prueba para el juego de las Letras. Requiere de: -un diccionario.txt -un letras.txt(con el formato dado en el guion) -el numero de letras a generar -modalidad . . . 74

Chapter 3

Class Documentation

3.1 bolsaLetras Class Reference

Classes

- class iterator

Public Member Functions

- **bolsaLetras** ()=default

  Constructor por defecto.
- bolsaLetras (const bolsaLetras &otra)

  Constructor de copia.
- bolsaLetras (const conjuntoLetras &c)

  Constructor por parámetros.
- int tama () const

  Devuelve el numero de letras en la bolsa.
- unordered_multiset< char > getBolsa ()

  Devuelve la bolsa.
- void aniadeLetra (const Letra &L)

  Añade un Objeto Letra a la bolsa.
- void aniadeLetra (const char &l)

  Añadir una letra a la bolsa.
- void quitaLetra (const Letra &L, bool todas)

  Elimina un Objeto Letra de la bolsa de letras.
- void quitaLetra (const char &l, bool todas)

  Elimina una letra de la bolsa de letras.
- bolsaLetras getLetras (int num)

  Devuelve una bolsa con un "num" de letras aleatorias de la bolsa actual.
- set< pair< int, string > > getSoluciones (const conjuntoLetras &Letras, const Diccionario &d, const char &modo)

  Obtiene todas las soluciones de la bolsa: Devuelve todas las palabras posibles con las letras de la bolsa y que existan en el diccionario. Además obtiene su puntuación según el modo de juego.
- bool Esta (const Letra &L)

  Comprueba si un Objeto Letra dado existe.

- bool Esta (const char &c)

  Comprueba si una letra dada existe.
- bool palabraValida (const string palabra)

  Comprueba si una palabra se puede formar con las letras de la bolsa.
- iterator begin ()

  Devuelve el iterador apuntando al inicio de la bolsa.
- iterator begin () const

  Devuelve el iterador apuntando al inicio de la bolsa.
- iterator end ()

  Devuelve el iterador apuntando al final de la bolsa.
- iterator end () const

  Devuelve el iterador apuntando al final de la bolsa.

Friends

- istream & operator>> (istream &is, bolsaLetras &bolsa)

  Lee de un flujo de entrada y lo escribe en un objeto Bolsa de Letras.
- ostream & operator<< (ostream &os, const bolsaLetras &bolsa)

  Escribe el objeto Bolsa de Letras en un flujo de salida.

3.1.1 Detailed Description

Definition at line 19 of file bolsaLetras.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 bolsaLetras() [1/2]

```
bolsaLetras::bolsaLetras (
            const bolsaLetras & otra )
```

Constructor de copia.

Parameters

otra	bolsaLetras a copiar

Definition at line 10 of file bolsaLetras.cpp.

```
00010                                           {
00011     bolsa = otra.bolsa;
00012 }
```

3.1.2.2 bolsaLetras() [2/2]

```
bolsaLetras::bolsaLetras (
            const conjuntoLetras & c )
```

Constructor por parámetros.

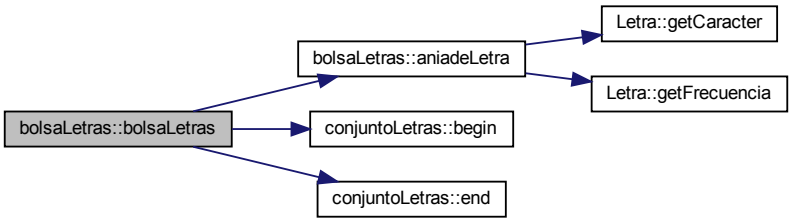
Parameters

c	conjunto de letras con el que llenar la bolsa

Definition at line 14 of file bolsaLetras.cpp.

```
00014                                                {
00015     conjuntoLetras::iterator it;
00016
00017     for (it = c.begin(); it != c.end(); ++it)
00018         aniadeLetra((*it));
00019
00020 }
```

Here is the call graph for this function:



3.1.3 Member Function Documentation

3.1.3.1 aniadeLetra() [1/2]

```
void bolsaLetras::aniadeLetra (
            const char & l )
```

Añadir una letra a la bolsa.

Parameters

L	Letra a añadir (a partir de un caracter)

Definition at line 40 of file bolsaLetras.cpp.

```
00040                                                {
00041     bolsa.insert(toupper(caracter)); //Lo convierte a mayuscula
00042 }
```

3.1.3.2 aniadeLetra() [2/2]

```
void bolsaLetras::aniadeLetra (
            const Letra & L )
```

Añade un Objeto Letra a la bolsa.

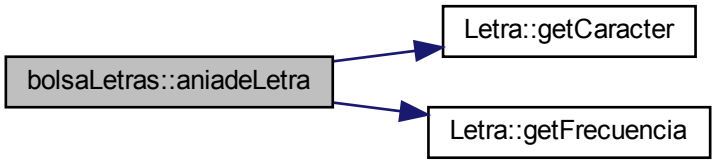
Parameters

L	Letra a añadir (a partir de un objeto del TDA Letra)

Definition at line 30 of file bolsaLetras.cpp.

```
00030                                           {
00031
00032     char caracter = L.getCaracter();
00033
00034     for(int i = 0; i < L.getFrecuencia(); i++){
00035         bolsa.insert(toupper(caracter)); //Lo convierte a mayuscula
00036     }
00037
00038 }
```

Here is the call graph for this function:



3.1.3.3 begin() [1/2]

```
bolsaLetras::iterator bolsaLetras::begin ( )
```

Devuelve el iterador apuntando al inicio de la bolsa.

Returns

iterador apuntando al inicio

Definition at line 139 of file bolsaLetras.cpp.

```
00139                                       {
00140     iterator iterador;
00141     iterador.it = bolsa.begin();
00142     return iterador;
00143
00144 }
```

3.1.3.4 begin() [2/2]

```
bolsaLetras::iterator bolsaLetras::begin ( ) const
```

Devuelve el iterador apuntando al inicio de la bolsa.

Returns

iterador apuntando al inicio

Definition at line 146 of file bolsaLetras.cpp.

```
00146                                           {
00147     iterator iterador;
00148     iterador.it = bolsa.begin();
00149     return iterador;
00150 }
```

3.1.3.5 end() [1/2]

```
bolsaLetras::iterator bolsaLetras::end ( )
```

Devuelve el iterador apuntando al final de la bolsa.

Returns

iterador apuntando al final

Definition at line 152 of file bolsaLetras.cpp.

```
00152                                         {
00153     iterator iterador;
00154     iterador.it = bolsa.end();
00155     return iterador;
00156 }
```

3.1.3.6 end() [2/2]

```
bolsaLetras::iterator bolsaLetras::end ( ) const
```

Devuelve el iterador apuntando al final de la bolsa.

Returns

iterador apuntando al final

Definition at line 158 of file bolsaLetras.cpp.

```
00158                                               {
00159     iterator iterador;
00160     iterador.it = bolsa.end();
00161     return iterador;
00162 }
```

3.1.3.7 Esta() [1/2]

```
bool bolsaLetras::Esta (
            const char & c )
```

Comprueba si una letra dada existe.

Parameters

c	Caracter a comprobar @bool true si existe el carácter, false en otro caso

Definition at line 98 of file bolsaLetras.cpp.

```
00098                                       {
00099     return ( bolsa.find(c) != bolsa.end()) ;
00100 }
```

3.1.3.8 Esta() [2/2]

```
bool bolsaLetras::Esta (
            const Letra & L )
```

Comprueba si un Objeto Letra dado existe.

Parameters

L	Objeto Letra a comprobar @bool true si existe el caracter que contiene el Objeto Letra, false en otro caso

Definition at line 94 of file bolsaLetras.cpp.

```
00094                                           {
00095     return ( bolsa.find(L.getCaracter()) != bolsa.end()) ;
00096 }
```

Here is the call graph for this function:

3.1.3.9 getBolsa()

```
unordered_multiset< char > bolsaLetras::getBolsa ( )
```

Devuelve la bolsa.

Returns

unordered_multiset<char> Bolsa de letras

Definition at line 26 of file bolsaLetras.cpp.

```
00026                                       {
00027     return bolsa;
00028 }
```

3.1.3.10 getLetras()

```
bolsaLetras bolsaLetras::getLetras (
            int num )
```

Devuelve una bolsa con un "num" de letras aleatorias de la bolsa actual.

Parameters

num	Tamaño de la bolsa a obtener

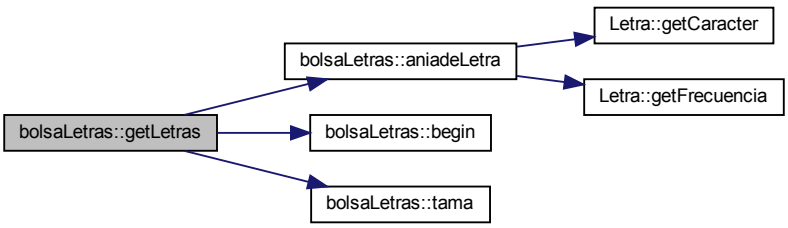
Returns

bolsaLetras con "num" de letras aleatorias

Definition at line 57 of file bolsaLetras.cpp.

```
00057                                           {
00058     bolsaLetras aux;
00059
00060     if (num < tama()){
00061         bolsaLetras::iterator it;
00062         int pos;
00063
00064         for (int i = 0; i < num; i++){
00065             it = begin();
00066             srand(time(NULL));
00067             pos = rand() % tama();
00068             for(int i = 0; i < pos; i++) //Avanza hasta pos (no podemos obtener el elemento con su
      indice)
00069                 ++it;
00070             aux.aniadeLetra((*it));
00071             it.it = bolsa.erase(it.it);
00072         }
00073     }
00074
00075     return aux;
00076 }
```

Here is the call graph for this function:



3.1.3.11 getSoluciones()

```
set< pair< int, string > > bolsaLetras::getSoluciones (
            const conjuntoLetras & Letras,
            const Diccionario & d,
            const char & modo )
```

Obtiene todas las soluciones de la bolsa: Devuelve todas las palabras posibles con las letras de la bolsa y que existan en el diccionario. Además obtiene su puntuación según el modo de juego.

Parameters

Letras	El Conjunto de Letras para obtener la puntuacion de cada palabra
d	Diccionario que contiene las palabras
modo	Modo de juego

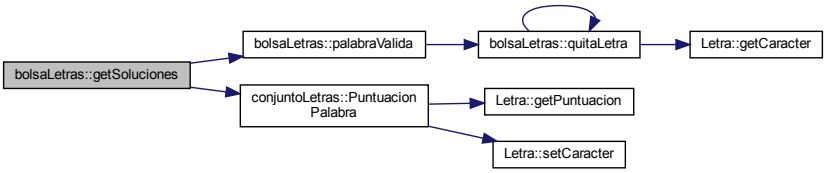
Returns

conjunto de pares, con la solucion, el primer elemento del pair es la puntuacion y el segundo la palabra

Definition at line 78 of file bolsaLetras.cpp.

```
00078
                    {
00079     set<pair<int,string» soluciones {};
00080     pair<int,string> sol;
00081     Diccionario::iterator it;
00082     for(it = d.begin(); it != d.end(); ++it){
00083         if(palabraValida((*it))){
00084             sol.second = (*it);
00085             sol.first = Letras.PuntuacionPalabra((*it), modo);
00086
00087             soluciones.insert(sol);
00088         }
00089     }
00090
00091     return soluciones;
00092 }
```

Here is the call graph for this function:



3.1.3.12 palabraValida()

```
bool bolsaLetras::palabraValida (
            const string palabra )
```

Comprueba si una palabra se puede formar con las letras de la bolsa.

Parameters

palabra	La palabra a comprobar

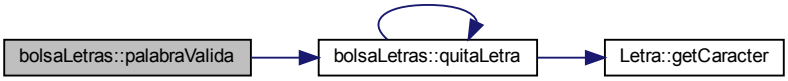
Returns

true si la palabra es válida según las reglas, false en otro caso

Definition at line 102 of file bolsaLetras.cpp.

```
00102                                                {
00103     bool esValida = true;
00104     bolsaLetras aux = *this;
00105     size_t i;
00106     for(i = 0; i < palabra.size() && esValida; i++){
00107         esValida = aux.bolsa.find(toupper(palabra.at(i))) != bolsa.end();
00108         if(esValida)
00109             aux.quitaLetra(toupper(palabra.at(i)),false); //Borra solo UNA instancia de esa letra
00110     }
00111     return esValida;
00112 }
```

Here is the call graph for this function:



3.1.3.13 quitaLetra() [1/2]

```
void bolsaLetras::quitaLetra (
            const char & l,
            bool todas )
```

Elimina una letra de la bolsa de letras.

Parameters

L	letra a eliminar (a partir de un caracter)

Definition at line 48 of file bolsaLetras.cpp.

```
00048                                                    {
00049     bolsaLetras::iterator it;
00050     it.it = bolsa.find(caracter);
00051     if(todas)
00052         bolsa.erase(*(it.it)); //Borra todas
00053     else
00054         bolsa.erase(it.it); //Borra solo 1
00055 }
```

3.1.3.14 quitaLetra() [2/2]

```
void bolsaLetras::quitaLetra (
            const Letra & L,
            bool todas )
```

Elimina un Objeto Letra de la bolsa de letras.

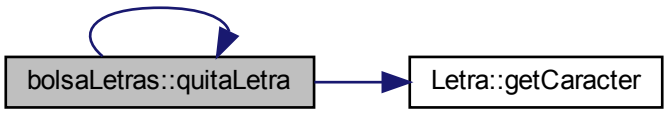
Parameters

L	Letra a eliminar (a partir de un objeto del TDA Letra)

Definition at line 44 of file bolsaLetras.cpp.

```
00044                                                   {
00045     quitaLetra(L.getCaracter(), todas);
00046 }
```

Here is the call graph for this function:



3.1.3.15 tama()

```
int bolsaLetras::tama ( ) const
```

Devuelve el numero de letras en la bolsa.

Definition at line 22 of file bolsaLetras.cpp.

```
00022                           {
00023     return bolsa.size();
00024 }
```

3.1.4 Friends And Related Function Documentation

3.1.4.1 operator<<

```
ostream & operator<< (
            ostream & os,
            const bolsaLetras & bolsa )  [friend]
```

Escribe el objeto Bolsa de Letras en un flujo de salida.

Parameters

os	flujo de salida
bolsa	el objeto bolsaLetras que se imprime

Returns

el flujo de salida

Definition at line 129 of file bolsaLetras.cpp.

```
00129                                                                {
00130     bolsaLetras::iterator it;
00131
00132     for (it = b.begin(); it != b.end(); ++it){
00133         os « (*it) « " ";
00134     }
00135
00136     return os;
00137 }
```

3.1.4.2 operator>>

```
istream & operator>> (
            istream & is,
            bolsaLetras & bolsa )  [friend]
```

Lee de un flujo de entrada y lo escribe en un objeto Bolsa de Letras.

Parameters

is	flujo de entrada
bolsa	objeto que recibe el flujo de entrada

Returns

el flujo de entrada

Definition at line 114 of file bolsaLetras.cpp.

```
00114                                                               {
00115     conjuntoLetras c;
00116
00117     is » c;
00118
00119     conjuntoLetras::iterator it;
00120
00121     for (it = c.begin(); it != c.end(); ++it){
00122         bolsa.aniadeLetra((*it));
00123     }
00124
00125     return is;
00126 }
```

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

- include/bolsaLetras.h
- src/bolsaLetras.cpp

3.2 conjuntoLetras Class Reference

Classes

- class iterator

Public Member Functions

- **conjuntoLetras** ()=default

  Constructor por defecto.
- int tama () const

  Devuelve el numero de letras que hay en el set<Letra>
- set< Letra > getLetras () const

  Devuelve el set de letras.
- void aniadeLetra (const Letra &Letra)

  Añade una Letra al set de letras.
- void quitaLetra (const Letra &Letra)

  Elimina una letra del set de letras.
- Letra getLetra (const int &i) const

  Devuelve una Letra del conjunto de letras.
- Letra getLetra (const char &c) const

  Devuelve una Letra del set de Letras.
- bool Esta (const Letra &Letra) const

  Comprobar si una letra pertenece al set de letras.
- int PuntuacionPalabra (const string palabra, const char &modo) const

  Calcula la puntuacion de una palabra, dado un modo de juego.
- int frecuenciaTotal () const

  Devuelve el total de las frecuencias de todas las letras del conjunto.
- conjuntoLetras contarLetras (const Diccionario &d)

  Cuenta las letras del diccinario pasado por parámetro.
- iterator begin ()

  Devuelve un iterador del TDA iterator, que apunta al inicio de la lista de palabras.
- iterator end ()

  Devuelve un iterador del TDA iterator, que apunta al fin de la lista de palabras.
- iterator begin () const

  Devuelve un iterador del TDA iterator, que apunta al inicio de la lista de palabras.
- iterator end () const

  Devuelve un iterador del TDA iterator, que apunta al fin de la lista de palabras.

Static Public Attributes

- static const string formato = "#Letra Frecuencia Puntos"

Friends

- istream & operator>> (istream &is, conjuntoLetras &conjunto)

  Lee de un flujo de entrada.
- ostream & operator<< (ostream &os, const conjuntoLetras &conjunto)

  Escribe en un flujo de salida el conjunto de Letras.

3.2.1 Detailed Description

Definition at line 19 of file conjuntoLetras.h.

3.2.2 Member Function Documentation

3.2.2.1 aniadeLetra()

```
void conjuntoLetras::aniadeLetra (
            const Letra & Letra )
```

Añade una Letra al set de letras.

Parameters

Letra	La Letra a añadir

Definition at line 20 of file conjuntoLetras.cpp.

```
00020                                                {
00021     Letras.insert(Letra);
00022 }
```

3.2.2.2 begin() [1/2]

```
conjuntoLetras::iterator conjuntoLetras::begin ( )
```

Devuelve un iterador del TDA iterator, que apunta al inicio de la lista de palabras.

Returns

iterator Iterador que apunta al inicio de la lista_palabras

Definition at line 124 of file conjuntoLetras.cpp.

```
00124                                               {
00125     iterator iterador;
00126     iterador.it = Letras.begin();
00127     return iterador;
00128 }
```

3.2.2.3 begin() [2/2]

```
conjuntoLetras::iterator conjuntoLetras::begin ( ) const
```

Devuelve un iterador del TDA iterator, que apunta al inicio de la lista de palabras.

Returns

iterator Iterador que apunta al inicio de la lista_palabras

Definition at line 130 of file conjuntoLetras.cpp.

```
00130                                                {
00131     iterator iterador;
00132     iterador.it = Letras.begin();
00133     return iterador;
00134
00135 }
```

3.2.2.4 contarLetras()

```
conjuntoLetras conjuntoLetras::contarLetras (
            const Diccionario & d )
```

Cuenta las letras del diccinario pasado por parámetro.

Parameters

d	Diccionario a contar

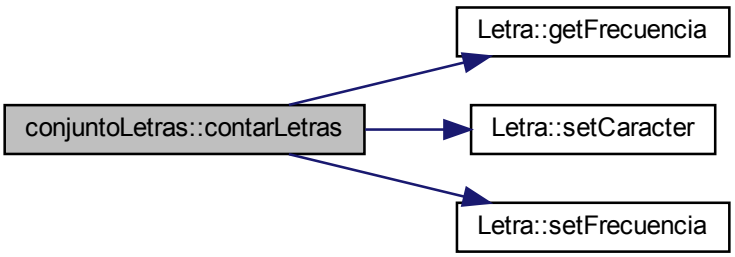
Returns

un objeto conjuntoLetras con la frecuencia de las letras

Definition at line 68 of file conjuntoLetras.cpp.

```
00068                                                                    {
00069     conjuntoLetras aux;
00070     Letra L;
00071
00072     Diccionario::iterator dit;
00073     set<Letra>::iterator sit;
00074     string palabra;
00075
00076     for(dit = d.begin(); dit != d.end(); ++dit){
00077         palabra = (*dit);
00078         for(size_t i=0; i < palabra.size(); i++){
00079             L.setCaracter(toupper(palabra.at(i))); //La convierte en mayuscula
00080             sit = Letras.find(L);
00081             if(sit != Letras.end()){ //Si la letra no existe, no hace nada
00082                 L = (*sit);
00083                 L.setFrecuencia(L.getFrecuencia()+1);
00084             }
00085         }
00086     }
00087     return aux;
00088 }
```

Here is the call graph for this function:



3.2.2.5 end() [1/2]

```
conjuntoLetras::iterator conjuntoLetras::end ( )
```

Devuelve un iterador del TDA iterator, que apunta al fin de la lista de palabras.

Returns

iterator Iterador que apunta al fin de la lista_palabras

Definition at line 137 of file conjuntoLetras.cpp.

```
00137                                          {
00138     iterator iterador;
00139     iterador.it = Letras.end();
00140     return iterador;
00141 }
```

3.2.2.6 end() [2/2]

```
conjuntoLetras::iterator conjuntoLetras::end ( ) const
```

Devuelve un iterador del TDA iterator, que apunta al fin de la lista de palabras.

Returns

iterator Iterador que apunta al fin de la lista_palabras

Definition at line 143 of file conjuntoLetras.cpp.

```
00143                                                {
00144     iterator iterador;
00145     iterador.it = Letras.end();
00146     return iterador;
00147 }
```

3.2.2.7 Esta()

```
bool conjuntoLetras::Esta (
            const Letra & Letra ) const
```

Comprobar si una letra pertenece al set de letras.

Parameters

letra	Letra a comprobar

Returns

true si la encuentra, false en otro caso

Definition at line 34 of file conjuntoLetras.cpp.

```
00034                                                   {
00035     return (Letras.find(Letra) != Letras.end());
00036 }
```

3.2.2.8 frecuenciaTotal()

```
int conjuntoLetras::frecuenciaTotal ( ) const
```

Devuelve el total de las frecuencias de todas las letras del conjunto.

Returns

la frecuencia total de letras

Definition at line 57 of file conjuntoLetras.cpp.

```
00057                                           {
00058     int total = 0;
00059
00060     set<Letra>::iterator it;
00061
00062     for (it = Letras.begin(); it != Letras.end(); ++it)
00063         total += (*it).getFrecuencia();
00064
00065     return total;
00066 }
```

3.2.2.9 getLetra() [1/2]

```
Letra conjuntoLetras::getLetra (
            const char & c ) const
```

Devuelve una Letra del set de Letras.

Parameters

c	Caracter con el que buscar la letra

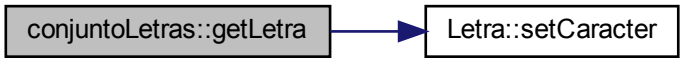
Returns

una Letra del set

Definition at line 28 of file conjuntoLetras.cpp.

```
00028                                                   {
00029     Letra L;
00030     L.setCaracter(c);
00031     return *(Letras.find(L));
00032 }
```

Here is the call graph for this function:



3.2.2.10 getLetra() [2/2]

```
Letra conjuntoLetras::getLetra (
            const int & i ) const
```

Devuelve una Letra del conjunto de letras.

Parameters

i	Posicion de la letra

Returns

la Letra del conjunto

3.2.2.11 getLetras()

```
set< Letra > conjuntoLetras::getLetras ( ) const
```

Devuelve el set de letras.

Returns

set<Letra> con las letras del conjunto

Definition at line 16 of file conjuntoLetras.cpp.

```
00016                                       {
00017     return Letras;
00018 }
```

3.2.2.12 PuntuacionPalabra()

```
int conjuntoLetras::PuntuacionPalabra (
            const string palabra,
            const char & modo ) const
```

Calcula la puntuacion de una palabra, dado un modo de juego.

Parameters

palabra	Palabra a calcular la puntuacion
modo	Mode de juego

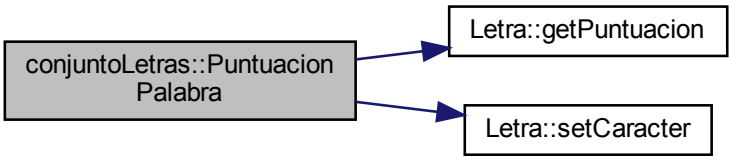
Returns

la puntuación de la palabra

Definition at line 38 of file conjuntoLetras.cpp.

```
00038                                                                                {
00039     int total = 0;
00040
00041     if (modo == 'P'){
00042         Letra L;
00043         for(size_t i = 0; i < palabra.size(); i++){
00044             L.setCaracter(toupper(palabra.at(i))); //Convierte la letra en mayusucla
00045             L = (*Letras.find(L));
00046
00047             total += L.getPuntuacion();
00048         }
00049     }
00050     else
00051         total = palabra.size();
00052
00053     return total;
00054 }
```

Here is the call graph for this function:



3.2.2.13 quitaLetra()

```
void conjuntoLetras::quitaLetra (
            const Letra & Letra )
```

Elimina una letra del set de letras.

Parameters

Letra	La Letra a eliminar

Definition at line 24 of file conjuntoLetras.cpp.

```
00024                                                   {
00025     Letras.erase(Letra);
00026 }
```

3.2.2.14 tama()

```
int conjuntoLetras::tama ( ) const
```

Devuelve el numero de letras que hay en el set<Letra>

Returns

El numero de letras

Definition at line 12 of file conjuntoLetras.cpp.

```
00012                                {
00013     return Letras.size();
00014 }
```

3.2.3 Friends And Related Function Documentation

3.2.3.1 operator<<

```
ostream & operator<< (
            ostream & os,
            const conjuntoLetras & conjunto )  [friend]
```

Escribe en un flujo de salida el conjunto de Letras.

Parameters

os	flujo de salida
conjunto	el objeto conjunto a escribir

Returns

el flujo de salida

Definition at line 111 of file conjuntoLetras.cpp.

```
00111                                                                {
00112
00113     os « conjunto.formato « endl;
00114
00115     conjuntoLetras::iterator it;
00116
00117     for(it = conjunto.begin(); it != conjunto.end(); ++it){
00118         os « (*it);
00119     }
00120     return os;
00121
00122 }
```

3.2.3.2 operator>>

```
istream & operator>> (
            istream & is,
            conjuntoLetras & conjunto )  [friend]
```

Lee de un flujo de entrada.

Parameters

is	flujo de entrada
conjunto	el objeto a ser leído

Returns

el flujo de entrada

Definition at line 91 of file conjuntoLetras.cpp.

```
00091                                                       {
00092
00093     Letra L;
00094     string primeraLinea;
00095
00096     getline(is, primeraLinea);
00097
00098     if(primeraLinea.empty() || primeraLinea != conjunto.formato){
00099         cerr « "ERROR: No se puede leer el conjunto de letras. La primera linea del "
00100              « "archivo letras.txt debe tener el formato: \n\t" « conjunto.formato « endl;
00101         exit(1);
00102     } else {
00103         while (is » L ){
00104             conjunto.Letras.insert(L);
00105         }
00106     }
00107
00108     return is;
00109 }
```

3.2.4 Member Data Documentation

3.2.4.1 formato

```
const string conjuntoLetras::formato = "#Letra Frecuencia Puntos"  [static]
```

Definition at line 24 of file conjuntoLetras.h.

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

- include/conjuntoLetras.h
- src/conjuntoLetras.cpp

3.3 Diccionario Class Reference

Classes

- class iterator

Public Member Functions

- **Diccionario** ()=default

  Construye un diccionario vacío.
- Diccionario (const Diccionario &diccionario)

  Constructor de copia.
- size_t size () const

  Consulta el tamaño del diccionario.
- bool Esta (const string &palabra) const

  Indica si una palabra está en el diccionario o no.
- vector< string > PalabrasLongitud (const size_t longitud) const

  Consulta cuántas palabras tienen dicha longitud.
- void aniade (const string &nueva_palabra)

  Añade una nueva palabra al diccionario.
- Diccionario & operator= (const Diccionario &diccionario)
- iterator begin ()
- iterator end ()
- iterator begin () const
- iterator end () const

Friends

- istream & operator>> (istream &is, Diccionario &diccionario)
- ostream & operator<< (ostream &os, const Diccionario &diccionario)
- ifstream & operator>> (ifstream &in, Diccionario &diccionario)

3.3.1 Detailed Description

Definition at line 19 of file diccionario.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Diccionario()

```
Diccionario::Diccionario (
            const Diccionario & diccionario )
```

Constructor de copia.

Definition at line 18 of file diccionario.cpp.

```
00018                                                   {
00019     *this = diccionario;
00020 }
```

3.3.3 Member Function Documentation

3.3.3.1 aniade()

```
void Diccionario::aniade (
            const string & nueva_palabra )
```

Añade una nueva palabra al diccionario.

Parameters

nueva_palabra	Palabra a añadir

Definition at line 40 of file diccionario.cpp.

```
00040                                                {
00041     datos.insert( nueva);
00042 }
```

3.3.3.2 begin() [1/2]

```
iterator Diccionario::begin ( )  [inline]
```

Definition at line 113 of file diccionario.h.

```
00113                     {
00114         iterator iter;
00115         iter.it = datos.begin();
00116         return iter;
00117     };
```

3.3.3.3 begin() [2/2]

```
iterator Diccionario::begin ( ) const  [inline]
```

Definition at line 125 of file diccionario.h.

```
00125                           {
00126         iterator iter;
00127         iter.it = datos.begin();
00128         return iter;
00129     };
```

3.3.3.4 end() [1/2]

```
iterator Diccionario::end ( )  [inline]
```

Definition at line 119 of file diccionario.h.

```
00119                   {
00120         iterator iter;
00121         iter.it = datos.end();
00122         return iter;
00123     };
```

3.3.3.5 end() [2/2]

```
iterator Diccionario::end ( ) const  [inline]
```

Definition at line 131 of file diccionario.h.

```
00131                         {
00132         iterator iter;
00133         iter.it = datos.end();
00134         return iter;
00135     };
```

3.3.3.6 Esta()

```
bool Diccionario::Esta (
            const string & palabra ) const
```

Indica si una palabra está en el diccionario o no.

Parameters

palabra	la palabra que se quiere buscar

Returns

true si la palabra está en el diccionario. False en caso contrario

Definition at line 26 of file diccionario.cpp.

```
00026                                                   {
00027     return datos.find( palabra) != datos.end();
00028 }
```

3.3.3.7 operator=()

```
Diccionario & Diccionario::operator= (
            const Diccionario & diccionario )
```

Definition at line 44 of file diccionario.cpp.

```
00044                                                       {
00045     datos = d.datos;
00046     return *this;
00047 }
```

3.3.3.8 PalabrasLongitud()

```
vector< string > Diccionario::PalabrasLongitud (
            const size_t longitud ) const
```

Consulta cuántas palabras tienen dicha longitud.

Returns

Vector con dicha cantidad de palabras

Definition at line 30 of file diccionario.cpp.

```
00030                                                                   {
00031     vector<string> resultado;
00032
00033     for(auto palabra: datos)
00034         if(palabra.size() == longitud)
00035             resultado.push_back(palabra);
00036
00037     return resultado;
00038 }
```

3.3.3.9 size()

```
size_t Diccionario::size ( ) const
```

Consulta el tamaño del diccionario.

Returns

Cantidad de palabras almacenadas

Definition at line 22 of file diccionario.cpp.

```
00022                                {
00023     return datos.size();
00024 }
```

3.3.4 Friends And Related Function Documentation

3.3.4.1 operator<<

```
ostream & operator<< (
            ostream & os,
            const Diccionario & diccionario )  [friend]
```

Definition at line 49 of file diccionario.cpp.

```
00049                                                                {
00050     int c = 1;
00051     for(auto palabra: diccionario){
00052         os « left « setw(25) «  to_string(c) + "-" + palabra;
00053         intercalado(c++,4);
00054     }
00055     cout « endl;
00056
00057     return os;
00058 }
```

3.3.4.2 operator>> [1/2]

```
ifstream & operator>> (
            ifstream & in,
            Diccionario & diccionario )  [friend]
```

Definition at line 71 of file diccionario.cpp.

```
00071                                                  { //Archivo de flujo de entrada
00072     string entrada;
00073
00074     while(getline(in, entrada))
00075         d.aniade(entrada);
00076
00077     return in;
00078 }
```

3.3.4.3 operator>> [2/2]

```
istream & operator>> (
            istream & is,
            Diccionario & diccionario )  [friend]
```

Definition at line 60 of file diccionario.cpp.

```
00060                                                { //flujo de entrada(ej. cin)
00061     string palabra;
00062
00063     is » palabra;
00064     d.aniade(palabra);
00065
00066     return is;
00067 }
```

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

- include/diccionario.h
- src/diccionario.cpp

3.4 bolsaLetras::iterator Class Reference

Public Member Functions

- **iterator** ()=default

  Constructor por defecto.
- char operator∗ ()

  Sobrecarga del operator∗.
- iterator & operator++ ()

  Sobrecarga del operator++.
- bool operator== (const iterator &i)

  Sobrecarga del operator==.
- bool operator!= (const iterator &i)

  Sobrecarga del operator!=.

Friends

- class bolsaLetras

3.4.1 Detailed Description

Definition at line 132 of file bolsaLetras.h.

3.4.2 Member Function Documentation

3.4.2.1 operator"!=()

```
bool bolsaLetras::iterator::operator!= (
            const iterator & i )
```

Sobrecarga del operator!=.

Parameters

i	El iterador a comparar con el actual

Returns

true si los iteratores no son iguales, false si lo son

Definition at line 171 of file bolsaLetras.cpp.

```
00171                                                          {
00172     return i.it != this->it;
00173 }
```

3.4.2.2 operator∗()

```
char bolsaLetras::iterator::operator* ( )
```

Sobrecarga del operator∗.

Returns

string Objeta Letra al que apunta el iterador

Definition at line 163 of file bolsaLetras.cpp.

```
00163                                  {
00164     return (*it);
00165 }
```

3.4.2.3 operator++()

```
bolsaLetras::iterator & bolsaLetras::iterator::operator++ ( )
```

Sobrecarga del operator++.

Returns

el iterador apuntando a la siguiente posición

Definition at line 175 of file bolsaLetras.cpp.

```
00175                                                  {
00176     ++it;
00177     return (*this);
00178 }
```

3.4.2.4 operator==()

```
bool bolsaLetras::iterator::operator== (
            const iterator & i )
```

Sobrecarga del operator==.

Parameters

i	El iterador a comparar con el actual

Returns

true si los iteratores son iguales, false si no lo son

Definition at line 167 of file bolsaLetras.cpp.

```
00167                                                    {
00168     return i.it == this->it;
00169 }
```

3.4.3 Friends And Related Function Documentation

3.4.3.1 bolsaLetras

```
friend class bolsaLetras  [friend]
```

Definition at line 167 of file bolsaLetras.h.

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

- include/bolsaLetras.h
- src/bolsaLetras.cpp

3.5 conjuntoLetras::iterator Class Reference

Public Member Functions

- **iterator** ()=default

  Constructor por defecto, crea un iterator vacio.
- Letra operator∗ ()

  Operador de acceso ∗.
- iterator & operator++ ()

  Operador ++, avanza en una posicion el iterador constante.
- bool operator== (const iterator &i)

  Operador ==, comprueba si un operador es igual a otro.
- bool operator!= (const iterator &i)

  Operador !=, comprueba si un operador es distinto a otro.

Friends

- class conjuntoLetras

3.5.1 Detailed Description

Definition at line 117 of file conjuntoLetras.h.

3.5.2 Member Function Documentation

3.5.2.1 operator"!=()

```
bool conjuntoLetras::iterator::operator!= (
            const iterator & i )
```

Operador !=, comprueba si un operador es distinto a otro.

Parameters

i	Iterador a comparar con el iterador implicito

Returns

bool Resultado de la comprobacion, true si se cumple, false si no

Definition at line 158 of file conjuntoLetras.cpp.

```
00158                                                                    {
00159     return i.it != it;
00160 }
```

3.5.2.2 operator∗()

```
Letra conjuntoLetras::iterator::operator* ( )
```

Operador de acceso ∗.

Returns

string Objeto al que apunta el iterador

Definition at line 162 of file conjuntoLetras.cpp.

```
00162                                    {
00163     return (*it);
00164 }
```

3.5.2.3 operator++()

```
conjuntoLetras::iterator & conjuntoLetras::iterator::operator++ ( )
```

Operador ++, avanza en una posicion el iterador constante.

Returns

iterator Iterados actual

Definition at line 149 of file conjuntoLetras.cpp.

```
00149                                                          {
00150     ++it;
00151     return (*this);
00152 }
```

3.5.2.4 operator==()

```
bool conjuntoLetras::iterator::operator== (
            const iterator & i )
```

Operador ==, comprueba si un operador es igual a otro.

Parameters

i	Iterador a comparar con el iterador implicito

Returns

bool Resultado de la comprobacion, true si se cumple, false si no

Definition at line 154 of file conjuntoLetras.cpp.

```
00154                                                                {
00155     return i.it == it;
00156 }
```

3.5.3 Friends And Related Function Documentation

3.5.3.1 conjuntoLetras

```
friend class conjuntoLetras  [friend]
```

Definition at line 160 of file conjuntoLetras.h.

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

- include/conjuntoLetras.h
- src/conjuntoLetras.cpp

3.6 Diccionario::iterator Class Reference

Public Member Functions

- string operator* ()
- iterator & operator++ ()
- bool operator== (const iterator &i)
- bool operator!= (const iterator &i)

Friends

- class Diccionario

3.6.1 Detailed Description

Definition at line 85 of file diccionario.h.

3.6.2 Member Function Documentation

3.6.2.1 operator"!=()

```
bool Diccionario::iterator::operator!= (
            const iterator & i )  [inline]
```

Definition at line 105 of file diccionario.h.

```
00105                                           {
00106             return this->it != i.it;
00107         }
```

3.6.2.2 operator*()

```
string Diccionario::iterator::operator* ( )  [inline]
```

Definition at line 92 of file diccionario.h.

```
00092                                 {
00093             return *(this->it);
00094         }
```

3.6.2.3 operator++()

```
iterator & Diccionario::iterator::operator++ ( )  [inline]
```

Definition at line 96 of file diccionario.h.

```
00096                                {
00097             ++this->it;
00098             return *this;
00099         }
```

3.6.2.4 operator==()

```
bool Diccionario::iterator::operator== (
             const iterator & i )  [inline]
```

Definition at line 101 of file diccionario.h.

```
00101                                                {
00102             return this->it == i.it;
00103         }
```

3.6.3 Friends And Related Function Documentation

3.6.3.1 Diccionario

```
friend class Diccionario  [friend]
```

Definition at line 109 of file diccionario.h.

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

- include/diccionario.h

3.7 Letra Class Reference

Public Member Functions

- Letra ()
  Constructor por defecto.
- Letra (const char &c, const int &frec, const int &punt)
  Constructor con parámetros.
- char getCaracter () const
  Devuelve el caracter asociado al objeto Letra.
- int getPuntuacion () const
  Devuelve la puntuacion asociada al objeto Letra.
- int getFrecuencia () const
  Obtener la cantidad asociada a una letra.
- void setCaracter (const char &c)
  Cambia el caracter actual(o vacio) por el caracter pasado por parámetro.
- void setFrecuencia (const int &frec)
  Cambia el frecuencia actual(o vacia) por la frecuencia pasada por parámetro.
- void setPuntuacion (const int &punt)
  Cambia la puntuación actual(o vacia) por la puntuación pasada por parámetro.
- bool operator== (const Letra &otra) const
  Sobrecarga del operator ==, comprueba si una letra es igual a otra.
- bool operator< (const Letra &otra) const
  Sobrecarga del operator <, comprueba si una letra es menor a otra.

Friends

- istream & operator>> (istream &is, Letra &L)

  Lee de un flujo de entrada de una Letra.
- ostream & operator<< (ostream &os, const Letra &L)

  Escribe en un flujo de salida una Letra.

3.7.1 Detailed Description

Definition at line 19 of file letra.h.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Letra() [1/2]

```
Letra::Letra ( )
```

Constructor por defecto.

Definition at line 9 of file letra.cpp.

```
00009             {
00010     caracter = 'A'; //Por defecto 'A'
00011     frecuencia = 0;
00012     puntuacion = 0;
00013 }
```

3.7.2.2 Letra() [2/2]

```
Letra::Letra (
            const char & c,
            const int & frec,
            const int & punt )
```

Constructor con parámetros.

Parameters

c	El caracter a añadir
frec	La frecuencia que aparece
punt	La puntuacion asignada a la letra

Definition at line 15 of file letra.cpp.

```
00015                                                   {
00016     caracter = c;
00017     frecuencia = frec;
00018     puntuacion = punt;
00019 }
```

3.7.3 Member Function Documentation

3.7.3.1 getCaracter()

```
char Letra::getCaracter ( ) const
```

Devuelve el caracter asociado al objeto Letra.

Returns

caracter El caracter del objeto Letra

Definition at line 21 of file letra.cpp.

```
00021                           {
00022     return caracter;
00023 }
```

3.7.3.2 getFrecuencia()

```
int Letra::getFrecuencia ( ) const
```

Obtener la cantidad asociada a una letra.

Returns

frecuencia La frecuencia con la que aparece el caracter

Definition at line 29 of file letra.cpp.

```
00029                               {
00030     return frecuencia;
00031 }
```

3.7.3.3 getPuntuacion()

```
int Letra::getPuntuacion ( ) const
```

Devuelve la puntuacion asociada al objeto Letra.

Returns

puntuacion La puntuacion asociada al caracter

Definition at line 25 of file letra.cpp.

```
00025                               {
00026     return puntuacion;
00027 }
```

3.7.3.4 operator<()

```
bool Letra::operator< (
            const Letra & otra ) const
```

Sobrecarga del operator <, comprueba si una letra es menor a otra.

Parameters

otra	Objeto Letra a comparar con la Letra actual

Returns

true si this mejor a otra, false en otro caso

Definition at line 49 of file letra.cpp.

```
00049                                       {
00050     return caracter < otra.getCaracter();
00051 }
```

Here is the call graph for this function:

3.7.3.5 operator==()

```
bool Letra::operator== (
            const Letra & otra ) const
```

Sobrecarga del operator ==, comprueba si una letra es igual a otra.

Parameters

otra	Objeto Letra a comparar con la Letra actual

Returns

true si son iguales, false en otro caso

Definition at line 45 of file letra.cpp.

```
00045                                       {
00046     return getCaracter() == otra.getCaracter();
00047 }
```

Here is the call graph for this function:

3.7.3.6 setCaracter()

```
void Letra::setCaracter (
            const char & c )
```

Cambia el caracter actual(o vacio) por el caracter pasado por parámetro.

Parameters

c	El carácter asociado a la Letra

Definition at line 33 of file letra.cpp.

```
00033                                       {
00034     caracter = c;
00035 }
```

3.7.3.7 setFrecuencia()

```
void Letra::setFrecuencia (
            const int & frec )
```

Cambia el frecuencia actual(o vacia) por la frecuencia pasada por parámetro.

Parameters

frec	La frecuencia asociada a la Letra

Definition at line 37 of file letra.cpp.

```
00037                                          {
00038     frecuencia = frec;
00039 }
```

3.7.3.8 setPuntuacion()

```
void Letra::setPuntuacion (
            const int & punt )
```

Cambia la puntuación actual(o vacia) por la puntuación pasada por parámetro.

Parameters

punt	La puntuación asociada a la Letra

Definition at line 41 of file letra.cpp.

```
00041                                           {
00042     puntuacion = punt;
00043 }
```

3.7.4 Friends And Related Function Documentation

3.7.4.1 operator<<

```
ostream & operator<< (
            ostream & os,
            const Letra & L )  [friend]
```

Escribe en un flujo de salida una Letra.

Parameters

os	el flujo de salida
L	Objeto Letra que se escribe

Returns

el flujo de salida

Definition at line 61 of file letra.cpp.

```
00061                                                 {
00062     os « L.caracter « "\t"
00063        « L.frecuencia « "\t"
00064        « L.puntuacion « endl;
00065
00066     return os;
00067 }
```

3.7.4.2 operator>>

```
istream & operator>> (
            istream & is,
            Letra & L )  [friend]
```

Lee de un flujo de entrada de una Letra.

Parameters

is	el flujo de entrada
L	Objeto Letra donde se realiza la lectura

Returns

el flujo de entrada

Definition at line 53 of file letra.cpp.

```
00053                                                {
00054     is » L.caracter;
00055     is » L.frecuencia;
00056     is » L.puntuacion;
00057
00058     return is;
00059 }
```

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

- include/letra.h
- src/letra.cpp

Chapter 4

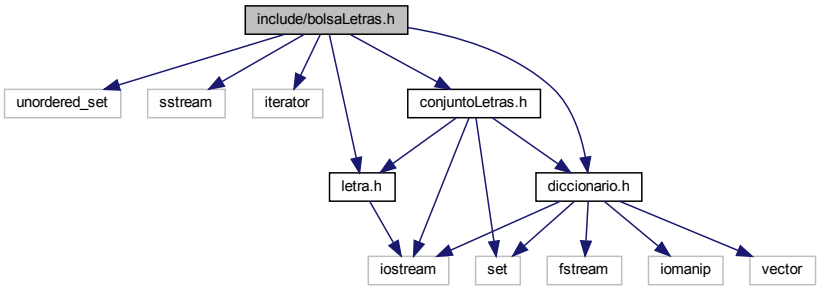
File Documentation

4.1 include/bolsaLetras.h File Reference

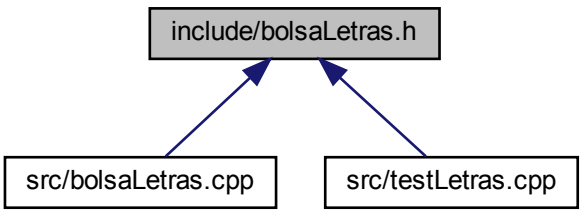
T.D.A. bolsaLetras.

```
#include <unordered_set>
#include <sstream>
#include <iterator>
#include "letra.h"
#include "conjuntoLetras.h"
#include "diccionario.h"
```

Include dependency graph for bolsaLetras.h:



This graph shows which files directly or indirectly include this file:



Classes

- class bolsaLetras
- class bolsaLetras::iterator

4.1.1 Detailed Description

T.D.A. bolsaLetras.

Author

Yeray Lopez Ramirez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file bolsaLetras.h.

4.2 bolsaLetras.h

Go to the documentation of this file.

```
00001 #ifndef __BOLSA_LETRAS_H__
00002 #define __BOLSA_LETRAS_H__
00003
00004 #include <unordered_set>
00005 #include <sstream>
00006 #include <iterator> //funcion advance
00007
00008 #include "letra.h"
00009 #include "conjuntoLetras.h"
00010 #include "diccionario.h"
00011
00019 class bolsaLetras{
00020     private:
00021         unordered_multiset<char> bolsa;
00022
00023     public:
00027         bolsaLetras() = default;
00028
00033         bolsaLetras(const bolsaLetras & otra);
00034
00039         bolsaLetras(const conjuntoLetras & c);
00040
00044         int tama() const;
00045
00050         unordered_multiset<char> getBolsa();
00051
00056         void aniadeLetra(const Letra & L);
00057
00062         void aniadeLetra(const char & l);
00063
00068         void quitaLetra(const Letra & L, bool todas);
00069
00074         void quitaLetra(const char & l, bool todas);
00075
00081         bolsaLetras getLetras(int num);
00082
00093         set<pair<int,string» getSoluciones(const conjuntoLetras & Letras, const Diccionario & d, const
      char & modo);
00094
00100         bool Esta(const Letra & L);
00101
00107         bool Esta(const char & c);
00108
00114         bool palabraValida(const string palabra);
```

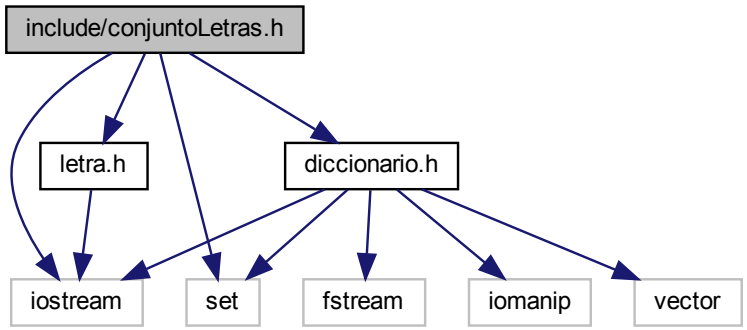
```
00115
00122         friend istream & operator» (istream &is, bolsaLetras &bolsa);
00123
00130         friend ostream & operator « (ostream &os, const bolsaLetras &bolsa);
00131
00132         class iterator {
00133         private:
00134             unordered_multiset<char>::const_iterator it; //el unordered_multiset usa iterador
      constante
00135         public:
00139                 iterator() = default;
00140
00145                 char operator* ();
00146
00151                 iterator & operator ++ ();
00152
00158                 bool operator ==(const iterator &i);
00159
00165                 bool operator !=(const iterator &i);
00166
00167                 friend class bolsaLetras;
00168         };
00169
00174         iterator begin();
00175
00180         iterator begin() const;
00181
00186         iterator end();
00187
00192         iterator end() const;
00193
00194 };
00195
00196 #endif
```

4.3 include/conjuntoLetras.h File Reference

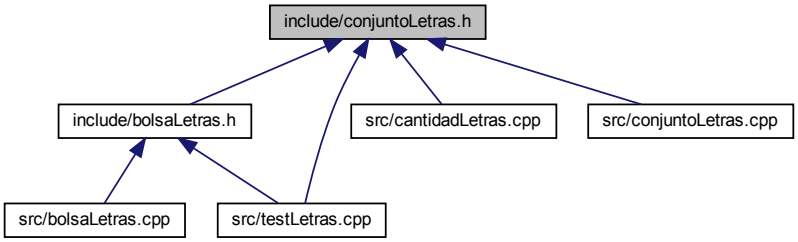
T.D.A Clase ConjuntoLetras que gestiona los archivos letra.txt.

```
#include <iostream>
#include <set>
#include "letra.h"
#include "diccionario.h"
```

Include dependency graph for conjuntoLetras.h:



This graph shows which files directly or indirectly include this file:



Classes

- class conjuntoLetras
- class conjuntoLetras::iterator

4.3.1 Detailed Description

T.D.A Clase ConjuntoLetras que gestiona los archivos letra.txt.

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file conjuntoLetras.h.

4.4 conjuntoLetras.h

Go to the documentation of this file.

```
00001 #ifndef __conjunto_letras_h__
00002 #define __conjunto_letras_h__
00003
00004 #include <iostream>
00005 #include <set>
00006
00007 #include "letra.h"
00008 #include "diccionario.h"
00009
00010 using namespace std;
00011
00019 class conjuntoLetras{
00020 private:
00021     set<Letra> Letras;
00022 public:
00023
00024     static const string formato; //La cabecera del archivo
00025
```

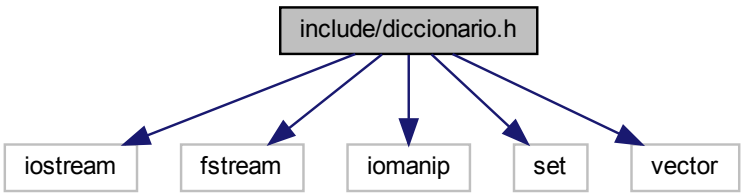
```
00029     conjuntoLetras() = default;
00030
00035     int tama() const;
00036
00041     set<Letra> getLetras() const;
00042
00047     void aniadeLetra(const Letra &Letra);
00048
00053     void quitaLetra(const Letra &Letra);
00054
00060     Letra getLetra(const int &i) const;
00061
00062
00068     Letra getLetra(const char & c) const;
00069
00070
00076     bool Esta(const Letra &Letra) const;
00077
00078
00085     int PuntuacionPalabra(const string palabra, const char & modo) const;
00086
00087
00092     int frecuenciaTotal() const;
00093
00099     conjuntoLetras contarLetras(const Diccionario &d);
00100
00107     friend istream & operator>> (istream &is, conjuntoLetras &conjunto);
00108
00115     friend ostream & operator <<(ostream &os, const conjuntoLetras &conjunto);
00116
00117     class iterator {
00118     private:
00119         set<Letra>::iterator it;
00120     public:
00125         iterator() = default;
00126
00133         Letra operator* ();
00134
00140         iterator & operator++ ();
00141
00149         bool operator==(const iterator &i);
00150
00158         bool operator !=(const iterator &i);
00159
00160         friend class conjuntoLetras;
00161         };
00162
00169         iterator begin();
00170
00177         iterator end();
00178
00185         iterator begin() const;
00186
00193         iterator end() const;
00194 };
00195
00196 #endif
```

4.5 include/diccionario.h File Reference

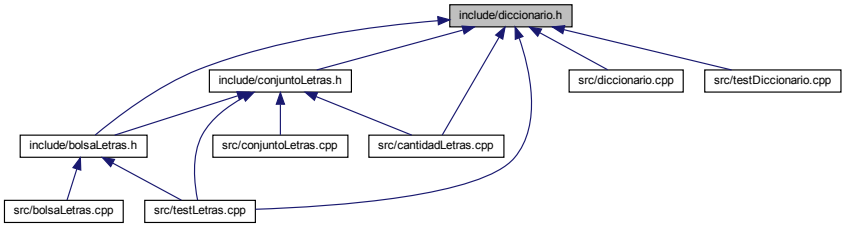
T.D.A Clase Diccionario.

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <set>
#include <vector>
```

Include dependency graph for diccionario.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Diccionario
- class Diccionario::iterator

Functions

- void intercalado (int c, int intercalar)

  Función externa para mejorar la salida. Hace un salto de linea cada "intercalar" veces.

4.5.1 Detailed Description

T.D.A Clase Diccionario.

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file diccionario.h.

4.5.2 Function Documentation

4.5.2.1 intercalado()

```
void intercalado (
            int c,
            int intercalar )
```

Función externa para mejorar la salida. Hace un salto de linea cada "intercalar" veces.

Parameters

c	Contador con el numero de impresiones hechas
intercalar	Cada cuantas impresiones se quiere dar el salto

Definition at line 13 of file diccionario.cpp.

```
00013                                           {
00014     if(c % intercalar == 0)
00015         cout « endl;
00016 }
```

4.6 diccionario.h

Go to the documentation of this file.

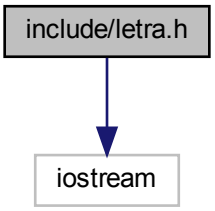
```
00001 #ifndef _DICCIONARIO_H
00002 #define _DICCIONARIO_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006 #include <iomanip>
00007 #include <set>
00008 #include <vector>
00009
00010 using namespace std;
00011
00019 class Diccionario {
00020 private:
00021
00022     set<string> datos;
00023
00024 public:
00025
00027     // CONSTRUCTORES
00029
00033     Diccionario() = default; //Se coge el por defecto
00034
00038     Diccionario( const Diccionario &diccionario );
00039
00041     //CONSULTA
00043
00048     size_t size() const; //Puede ser int tb pero es mas "correcto" size_t
00049
00055     bool Esta(const string & palabra) const;
00056
00061     vector<string> PalabrasLongitud(const size_t longitud) const;
00062
00064     //MODIFICADORES
00066
00071     void aniade(const string& nueva_palabra );
00072
00074     //OPERADORES
00076
00077     Diccionario & operator= (const Diccionario& diccionario);
00078     friend istream & operator» (istream &is, Diccionario &diccionario);
```

```
00079     friend ostream & operator« (ostream &os, const Diccionario &diccionario);
00080     friend ifstream & operator» (ifstream &in, Diccionario &diccionario);
00082     //ITERADORES
00084
00085     class iterator{
00086     private:
00087         set<string>::iterator it;
00088     public:
00089
00090         iterator() = default; //Nada que tocar, por defecto
00091
00092         string operator *(){
00093             return *(this->it);
00094         }
00095
00096         iterator & operator++(){
00097             ++this->it;
00098             return *this;
00099         }
00100
00101         bool operator ==(const iterator &i){
00102             return this->it == i.it;
00103         }
00104
00105         bool operator !=(const iterator &i){
00106             return this->it != i.it;
00107         }
00108
00109         friend class Diccionario;
00110
00111     };
00112
00113     iterator begin(){
00114         iterator iter;
00115         iter.it = datos.begin();
00116         return iter;
00117     };
00118
00119     iterator end(){
00120         iterator iter;
00121         iter.it = datos.end();
00122         return iter;
00123     };
00124
00125     iterator begin() const{
00126         iterator iter;
00127         iter.it = datos.begin();
00128         return iter;
00129     };
00130
00131     iterator end() const{
00132         iterator iter;
00133         iter.it = datos.end();
00134         return iter;
00135     };
00136
00137 };
00138
00140 //FUNCION EXTERNA
00142
00148 void intercalado(int c, int intercalar);
00149
00150 #endif
```

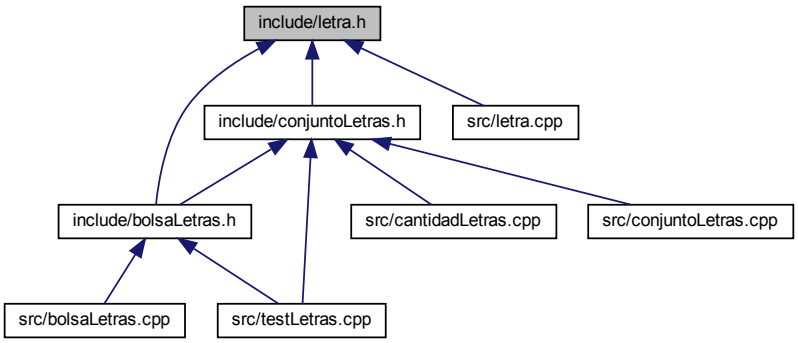
4.7 include/letra.h File Reference

T.D.A. Clase Letra Clase letra que representa la unidad mínima del juego. Formada por: Caracter Frecuencia Puntuacion.

`#include <iostream>`
Include dependency graph for letra.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Letra

4.7.1 Detailed Description

T.D.A. Clase Letra Clase letra que representa la unidad mínima del juego. Formada por: Caracter Frecuencia Puntuacion.

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file letra.h.

4.8 letra.h

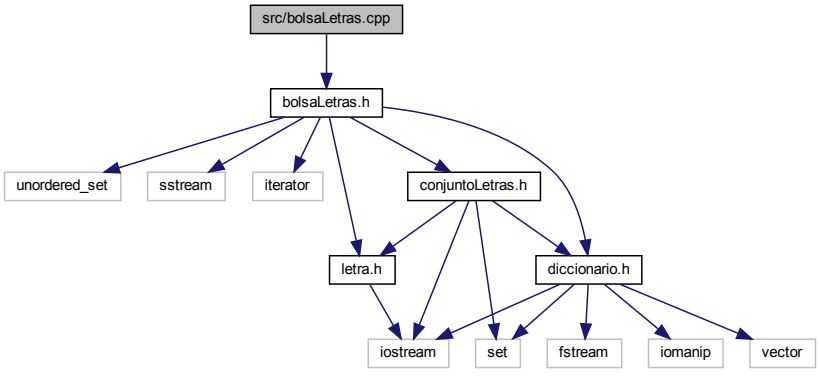
Go to the documentation of this file.

```
00001 #ifndef __LETRA_H__
00002 #define __LETRA_H__
00003
00004 #include <iostream>
00005
00006 using namespace std;
00007
00008
00019 class Letra{
00020 private:
00021     char caracter;
00022     int frecuencia;
00023     int puntuacion;
00024 public:
00025
00029     Letra();
00030
00037     Letra(const char &c, const int &frec, const int &punt);
00038
00043     char getCaracter() const;
00044
00049     int getPuntuacion () const;
00050
00055     int getFrecuencia() const;
00056
00061     void setCaracter(const char &c);
00062
00067     void setFrecuencia(const int &frec);
00068
00073     void setPuntuacion(const int &punt);
00074
00080     bool operator== (const Letra & otra) const;
00081
00087     bool operator< (const Letra & otra) const;
00088
00089
00096     friend istream & operator» (istream &is, Letra &L);
00097
00098
00105     friend ostream & operator« (ostream &os, const Letra &L);
00106
00107 };
00108
00109 #endif
```

4.9 src/bolsaLetras.cpp File Reference

`#include "bolsaLetras.h"`

Include dependency graph for bolsaLetras.cpp:



Functions

- istream & operator>> (istream &is, bolsaLetras &bolsa)
- ostream & operator<< (ostream &os, const bolsaLetras &b)

4.9.1 Detailed Description

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file bolsaLetras.cpp.

4.9.2 Function Documentation

4.9.2.1 operator<<()

```
ostream & operator<< (
            ostream & os,
            const bolsaLetras & b )
```

Parameters

os	flujo de salida
bolsa	el objeto bolsaLetras que se imprime

Returns

el flujo de salida

Definition at line 129 of file bolsaLetras.cpp.

```
00129                                                   {
00130     bolsaLetras::iterator it;
00131
00132     for (it = b.begin(); it != b.end(); ++it){
00133         os « (*it) « " ";
00134     }
00135
00136     return os;
00137 }
```

4.9.2.2 operator>>()

```
istream & operator>> (
            istream & is,
            bolsaLetras & bolsa )
```

Parameters

is	flujo de entrada
bolsa	objeto que recibe el flujo de entrada

Returns

el flujo de entrada

Definition at line 114 of file bolsaLetras.cpp.

```
00114                                                    {
00115     conjuntoLetras c;
00116
00117     is » c;
00118
00119     conjuntoLetras::iterator it;
00120
00121     for (it = c.begin(); it != c.end(); ++it){
00122         bolsa.aniadeLetra((*it));
00123     }
00124
00125     return is;
00126 }
```

4.10 bolsaLetras.cpp

Go to the documentation of this file.

```
00001
00008 #include "bolsaLetras.h"
00009
00010 bolsaLetras::bolsaLetras(const bolsaLetras &otra){
00011     bolsa = otra.bolsa;
00012 }
00013
00014 bolsaLetras::bolsaLetras(const conjuntoLetras & c){
00015     conjuntoLetras::iterator it;
00016
00017     for (it = c.begin(); it != c.end(); ++it)
00018         aniadeLetra((*it));
00019
00020 }
00021
00022 int bolsaLetras::tama() const{
00023     return bolsa.size();
00024 }
00025
00026 unordered_multiset<char> bolsaLetras::getBolsa(){
00027     return bolsa;
00028 }
00029
00030 void bolsaLetras::aniadeLetra(const Letra & L){
00031
00032     char caracter = L.getCaracter();
00033
00034     for(int i = 0; i < L.getFrecuencia(); i++){
00035         bolsa.insert(toupper(caracter)); //Lo convierte a mayuscula
00036     }
00037
00038 }
00039
00040 void bolsaLetras::aniadeLetra(const char & caracter){
00041     bolsa.insert(toupper(caracter)); //Lo convierte a mayuscula
00042 }
00043
```

```
00044 void bolsaLetras::quitaLetra(const Letra & L, bool todas){
00045     quitaLetra(L.getCaracter(), todas);
00046 }
00047
00048 void bolsaLetras::quitaLetra(const char & caracter, bool todas){
00049     bolsaLetras::iterator it;
00050     it.it = bolsa.find(caracter);
00051     if(todas)
00052         bolsa.erase(*(it.it)); //Borra todas
00053     else
00054         bolsa.erase(it.it); //Borra solo 1
00055 }
00056
00057 bolsaLetras bolsaLetras::getLetras(int num){
00058     bolsaLetras aux;
00059
00060     if (num < tama()){
00061         bolsaLetras::iterator it;
00062         int pos;
00063
00064         for (int i = 0; i < num; i++){
00065             it = begin();
00066             srand(time(NULL));
00067             pos = rand() % tama();
00068             for(int i = 0; i < pos; i++) //Avanza hasta pos (no podemos obtener el elemento con su
      indice)
00069                 ++it;
00070             aux.aniadeLetra((*it));
00071             it.it = bolsa.erase(it.it);
00072         }
00073     }
00074
00075     return aux;
00076 }
00077
00078 set<pair<int,string» bolsaLetras::getSoluciones(const conjuntoLetras & Letras, const Diccionario & d,
      const char & modo){
00079     set<pair<int,string» soluciones {};
00080     pair<int,string> sol;
00081     Diccionario::iterator it;
00082     for(it = d.begin(); it != d.end(); ++it){
00083         if(palabraValida((*it))){
00084             sol.second = (*it);
00085             sol.first = Letras.PuntuacionPalabra((*it), modo);
00086
00087             soluciones.insert(sol);
00088         }
00089     }
00090
00091     return soluciones;
00092 }
00093
00094 bool bolsaLetras::Esta(const Letra & L){
00095     return ( bolsa.find(L.getCaracter()) != bolsa.end()) ;
00096 }
00097
00098 bool bolsaLetras::Esta(const char & c){
00099     return ( bolsa.find(c) != bolsa.end()) ;
00100 }
00101
00102 bool bolsaLetras::palabraValida(const string palabra){
00103     bool esValida = true;
00104     bolsaLetras aux = *this;
00105     size_t i;
00106     for(i = 0; i < palabra.size() && esValida; i++){
00107         esValida = aux.bolsa.find(toupper(palabra.at(i))) != bolsa.end();
00108         if(esValida)
00109             aux.quitaLetra(toupper(palabra.at(i)),false); //Borra solo UNA instancia de esa letra
00110     }
00111     return esValida;
00112 }
00113
00114 istream & operator» (istream &is, bolsaLetras &bolsa){
00115     conjuntoLetras c;
00116
00117     is » c;
00118
00119     conjuntoLetras::iterator it;
00120
00121     for (it = c.begin(); it != c.end(); ++it){
00122         bolsa.aniadeLetra((*it));
00123     }
00124
00125     return is;
00126 }
00127
00128
```

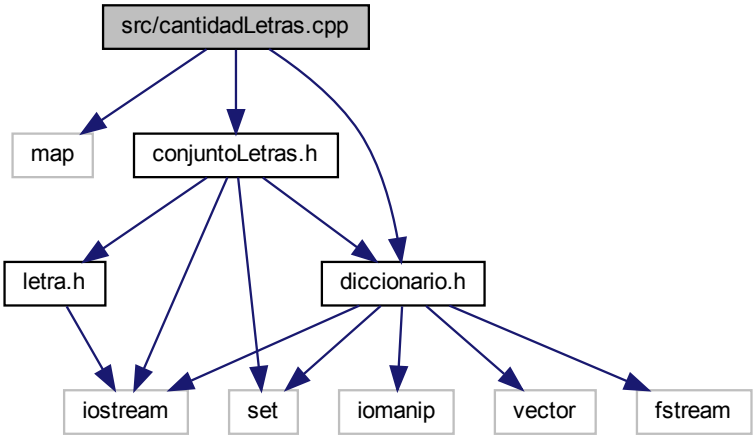
```
00129 ostream & operator « (ostream &os, const bolsaLetras &b){
00130     bolsaLetras::iterator it;
00131
00132     for (it = b.begin(); it != b.end(); ++it){
00133         os « (*it) « " ";
00134     }
00135
00136     return os;
00137 }
00138
00139 bolsaLetras::iterator bolsaLetras::begin(){
00140     iterator iterador;
00141     iterador.it = bolsa.begin();
00142     return iterador;
00143
00144 }
00145
00146 bolsaLetras::iterator bolsaLetras::begin() const{
00147     iterator iterador;
00148     iterador.it = bolsa.begin();
00149     return iterador;
00150 }
00151
00152 bolsaLetras::iterator bolsaLetras::end(){
00153     iterator iterador;
00154     iterador.it = bolsa.end();
00155     return iterador;
00156 }
00157
00158 bolsaLetras::iterator bolsaLetras::end() const{
00159     iterator iterador;
00160     iterador.it = bolsa.end();
00161     return iterador;
00162 }
00163 char bolsaLetras::iterator::operator* (){
00164     return (*it);
00165 }
00166
00167 bool bolsaLetras::iterator::operator==(const bolsaLetras::iterator &i){
00168     return i.it == this->it;
00169 }
00170
00171 bool bolsaLetras::iterator::operator!=(const bolsaLetras::iterator &i){
00172     return i.it != this->it;
00173 }
00174
00175 bolsaLetras::iterator & bolsaLetras::iterator::operator++(){
00176     ++it;
00177     return (*this);
00178 }
```

4.11 src/cantidadLetras.cpp File Reference

Programa de prueba donde a partir de un diccionario y un conjunto de letras, genera un archivo de salida con las frecuencias absolutas y relativas de cada letra del diccionario. Ademas actualiza las cantidades y puntuaciones del archivo conjunto de letras(letras.txt).

```
#include <map>
#include "diccionario.h"
#include "conjuntoLetras.h"
```

Include dependency graph for cantidadLetras.cpp:



Functions

- int main (int argc, char **argv)

4.11.1 Detailed Description

Programa de prueba donde a partir de un diccionario y un conjunto de letras, genera un archivo de salida con las frecuencias absolutas y relativas de cada letra del diccionario. Ademas actualiza las cantidades y puntuaciones del archivo conjunto de letras(letras.txt).

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file cantidadLetras.cpp.

4.11.2 Function Documentation

4.11.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

Definition at line 18 of file cantidadLetras.cpp.

```
00019 {
00020     if (argc != 4)
00021     {
00022         cout « "Uso: cantidad_letras <diccionario.txt> <letras.txt> <ficherosalida.txt>" « endl;
00023         exit(EXIT_SUCCESS);
00024     }
00025
00027     //CARGAR FICHERO DICCIONARIO.txt y LETRAS.txt
00029     fstream f(argv[1]);
00030     Diccionario D;
00031     conjuntoLetras Letras;
00032
00033     if (!f)
00034     {
00035         cerr « "Error: No se puede abrir el fichero diccionario" « endl;
00036         return 0;
00037     }
00038     while(!f.eof())
00039         f » D;
00040     f.close();
00041
00042     f.open(argv[2]);
00043     if (!f)
00044     {
00045         cerr « "Error: No se puede abrir el fichero de letras" « endl;
00046         return 0;
00047     }
00048     f » Letras;
00049     f.close();
00050
00052     //CALCULO DE APARICIONES
00054
00055     map<char,int> frecsAbsolutas;
00056     map<char,int>::iterator itMap;
00057     double  frecTotal = 0;
00058     string palabra;
00059
00060     for (conjuntoLetras::iterator it = Letras.begin(); it != Letras.end(); ++it)
00061         frecsAbsolutas.insert(make_pair((*it).getCaracter(), 0));
00062
00063     for (Diccionario::iterator itDic = D.begin(); itDic != D.end(); ++itDic)
00064     {
00065         palabra = (*itDic);
00066         for (size_t i = 0; i < palabra.size(); i++){
00067             itMap = frecsAbsolutas.find(toupper(palabra.at(i)));
00068             if(itMap != frecsAbsolutas.end())
00069             {
00070                 (*itMap).second++;
00071                 frecTotal++;
00072             }
00073         }
00074     }
00075
00077     //ESCRITURA DE LOS RESULTADOS EN LOS ARCHIVOS
00079
00080     char car;
00081     double frecRelativa;
00082     fstream fLetras(argv[2], fstream::out | fstream::trunc);
00083
00084     if (!fLetras)
00085     {
00086         cerr « "Error: No se puede abrir el fichero de letras" « endl;
00087         return 0;
00088     }
00089
00090     f.open(argv[3], fstream::out | fstream::trunc);
00091     if (!f)
00092     {
00093         cerr « "Error: No se puede abrir el fichero de salida" « endl;
00094         return 0;
00095     }
00096
00097     fLetras « Letras.formato « endl;
00098     f « "#Letra    FAbs.   FRel." « endl;
00099
00100     int factorCorrPuntos = 5; //Las primeras letras tendran menos puntuacion a pesar de aparecer poco
00101     for (conjuntoLetras::iterator it = Letras.begin(); it != Letras.end(); ++it)
```

```
00102     {
00103         car = (*it).getCaracter();
00104         frecRelativa = frecsAbsolutas[car] / frecTotal;
00105 
00106         f « car « "\t" « frecsAbsolutas[car] « "\t" « frecRelativa « "\n";
00107 
00108         frecRelativa *= 100;
00109 
00110         if (frecRelativa < 1)
00111             fLetras « car « "\t" « 1 « "\t";
00112         else
00113             fLetras « car « "\t" « (int) frecRelativa « "\t";
00114 
00115         if (10/frecRelativa > 10)
00116             if(factorCorrPuntos > -1)
00117                 fLetras « (10 - factorCorrPuntos--) « endl;
00118             else fLetras « 10 « endl;
00119         else if (10/frecRelativa < 1)
00120             fLetras « 1 « endl;
00121         else
00122             fLetras « (int) (10/frecRelativa) « endl;
00123     }
00124     fLetras.close();
00125     f.close();
00126 }
```

4.12 cantidadLetras.cpp

Go to the documentation of this file.

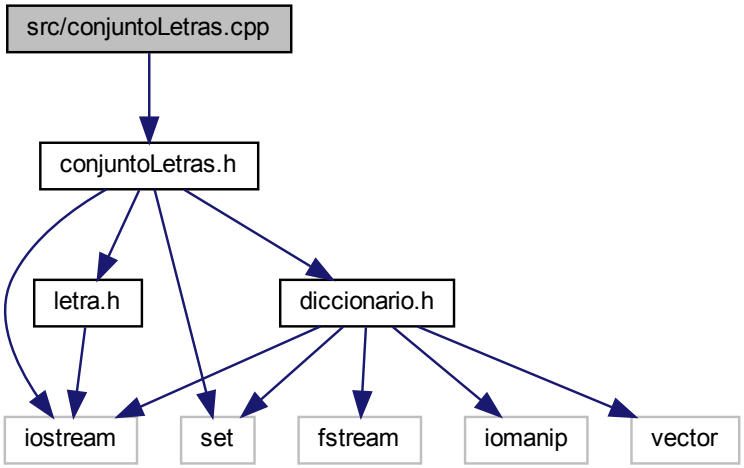
```
00001 #include <map>
00002 
00003 #include "diccionario.h"
00004 #include "conjuntoLetras.h"
00005 
00006 using namespace std;
00007 
00018 int main(int argc, char **argv)
00019 {
00020     if (argc != 4)
00021     {
00022         cout « "Uso: cantidad_letras <diccionario.txt> <letras.txt> <ficherosalida.txt>" « endl;
00023         exit(EXIT_SUCCESS);
00024     }
00025 
00027     //CARGAR FICHERO DICCIONARIO.txt y LETRAS.txt
00029     fstream f(argv[1]);
00030     Diccionario D;
00031     conjuntoLetras Letras;
00032 
00033     if (!f)
00034     {
00035         cerr « "Error: No se puede abrir el fichero diccionario" « endl;
00036         return 0;
00037     }
00038     while(!f.eof())
00039         f » D;
00040     f.close();
00041 
00042     f.open(argv[2]);
00043     if (!f)
00044     {
00045         cerr « "Error: No se puede abrir el fichero de letras" « endl;
00046         return 0;
00047     }
00048     f » Letras;
00049     f.close();
00050 
00052     //CALCULO DE APARICIONES
00054 
00055     map<char,int> frecsAbsolutas;
00056     map<char,int>::iterator itMap;
00057     double  frecTotal = 0;
00058     string palabra;
00059 
00060     for (conjuntoLetras::iterator it = Letras.begin(); it != Letras.end(); ++it)
00061         frecsAbsolutas.insert(make_pair((*it).getCaracter(), 0));
00062 
00063     for (Diccionario::iterator itDic = D.begin(); itDic != D.end(); ++itDic)
00064     {
00065         palabra = (*itDic);
00066         for (size_t i = 0; i < palabra.size(); i++){
```

```
00067             itMap = frecsAbsolutas.find(toupper(palabra.at(i)));
00068             if(itMap != frecsAbsolutas.end())
00069             {
00070                 (*itMap).second++;
00071                 frecTotal++;
00072             }
00073         }
00074     }
00075
00077     //ESCRITURA DE LOS RESULTADOS EN LOS ARCHIVOS
00079
00080     char car;
00081     double frecRelativa;
00082     fstream fLetras(argv[2], fstream::out | fstream::trunc);
00083
00084     if (!fLetras)
00085     {
00086         cerr « "Error: No se puede abrir el fichero de letras" « endl;
00087         return 0;
00088     }
00089
00090     f.open(argv[3], fstream::out | fstream::trunc);
00091     if (!f)
00092     {
00093         cerr « "Error: No se puede abrir el fichero de salida" « endl;
00094         return 0;
00095     }
00096
00097     fLetras « Letras.formato « endl;
00098     f « "#Letra    FAbs.   FRel." « endl;
00099
00100     int factorCorrPuntos = 5; //Las primeras letras tendran menos puntuacion a pesar de aparecer poco
00101     for (conjuntoLetras::iterator it = Letras.begin(); it != Letras.end(); ++it)
00102     {
00103         car = (*it).getCaracter();
00104         frecRelativa = frecsAbsolutas[car] / frecTotal;
00105
00106         f « car « "\t" « frecsAbsolutas[car] « "\t" « frecRelativa « "\n";
00107
00108         frecRelativa *= 100;
00109
00110         if (frecRelativa < 1)
00111             fLetras « car « "\t" « 1 « "\t";
00112         else
00113             fLetras « car « "\t" « (int) frecRelativa « "\t";
00114
00115         if (10/frecRelativa > 10)
00116             if(factorCorrPuntos > -1)
00117                 fLetras « (10 - factorCorrPuntos--) « endl;
00118             else fLetras « 10 « endl;
00119         else if (10/frecRelativa < 1)
00120             fLetras « 1 « endl;
00121         else
00122             fLetras « (int) (10/frecRelativa) « endl;
00123     }
00124     fLetras.close();
00125     f.close();
00126 }
```

4.13 src/conjuntoLetras.cpp File Reference

```
#include "conjuntoLetras.h"
```

Include dependency graph for conjuntoLetras.cpp:



Functions

- istream & operator>> (istream &is, conjuntoLetras &conjunto)
- ostream & operator<< (ostream &os, const conjuntoLetras &conjunto)

4.13.1 Detailed Description

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file conjuntoLetras.cpp.

4.13.2 Function Documentation

4.13.2.1 operator<<()

```
ostream & operator<< (
            ostream & os,
            const conjuntoLetras & conjunto )
```

Parameters

os	flujo de salida
conjunto	el objeto conjunto a escribir

Returns

el flujo de salida

Definition at line 111 of file conjuntoLetras.cpp.

```
00111                                                                {
00112
00113     os « conjunto.formato « endl;
00114
00115     conjuntoLetras::iterator it;
00116
00117     for(it = conjunto.begin(); it != conjunto.end(); ++it){
00118         os « (*it);
00119     }
00120     return os;
00121
00122 }
```

4.13.2.2 operator>>()

```
istream & operator>> (
            istream & is,
            conjuntoLetras & conjunto )
```

Parameters

is	flujo de entrada
conjunto	el objeto a ser leído

Returns

el flujo de entrada

Definition at line 91 of file conjuntoLetras.cpp.

```
00091                                                          {
00092
00093     Letra L;
00094     string primeraLinea;
00095
00096     getline(is, primeraLinea);
00097
00098     if(primeraLinea.empty() || primeraLinea != conjunto.formato){
00099         cerr « "ERROR: No se puede leer el conjunto de letras. La primera linea del "
00100              « "archivo letras.txt debe tener el formato: \n\t" « conjunto.formato « endl;
00101         exit(1);
00102     } else {
00103         while (is » L ){
00104             conjunto.Letras.insert(L);
00105         }
00106     }
00107
00108     return is;
00109 }
```

4.14 conjuntoLetras.cpp

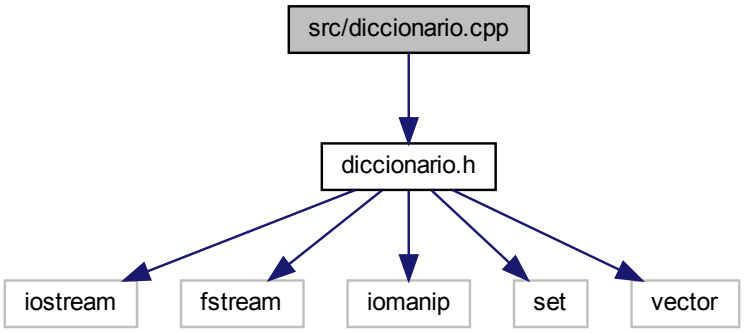
Go to the documentation of this file.

```
00001
00008 #include "conjuntoLetras.h"
00009
00010 const string conjuntoLetras::formato = "#Letra Frecuencia Puntos";
00011
00012 int conjuntoLetras::tama() const{
00013     return Letras.size();
00014 }
00015
00016 set<Letra> conjuntoLetras::getLetras() const{
00017     return Letras;
00018 }
00019
00020 void conjuntoLetras::aniadeLetra(const Letra &Letra){
00021     Letras.insert(Letra);
00022 }
00023
00024 void conjuntoLetras::quitaLetra(const Letra &Letra){
00025     Letras.erase(Letra);
00026 }
00027
00028 Letra conjuntoLetras::getLetra(const char &c) const{
00029     Letra L;
00030     L.setCaracter(c);
00031     return *(Letras.find(L));
00032 }
00033
00034 bool conjuntoLetras::Esta(const Letra &Letra) const{
00035     return (Letras.find(Letra) != Letras.end());
00036 }
00037
00038 int conjuntoLetras::PuntuacionPalabra(const string palabra, const char & modo) const{
00039     int total = 0;
00040
00041     if (modo == 'P'){
00042         Letra L;
00043         for(size_t i = 0; i < palabra.size(); i++){
00044             L.setCaracter(toupper(palabra.at(i))); //Convierte la letra en mayusucla
00045             L = (*Letras.find(L));
00046
00047             total += L.getPuntuacion();
00048         }
00049     }
00050     else
00051         total = palabra.size();
00052
00053     return total;
00054 }
00055
00056
00057 int conjuntoLetras::frecuenciaTotal() const{
00058     int total = 0;
00059
00060     set<Letra>::iterator it;
00061
00062     for (it = Letras.begin(); it != Letras.end(); ++it)
00063         total += (*it).getFrecuencia();
00064
00065     return total;
00066 }
00067
00068 conjuntoLetras conjuntoLetras::contarLetras(const Diccionario &d){
00069     conjuntoLetras aux;
00070     Letra L;
00071
00072     Diccionario::iterator dit;
00073     set<Letra>::iterator sit;
00074     string palabra;
00075
00076     for(dit = d.begin(); dit != d.end(); ++dit){
00077         palabra = (*dit);
00078         for(size_t i=0; i < palabra.size(); i++){
00079             L.setCaracter(toupper(palabra.at(i))); //La convierte en mayuscula
00080             sit = Letras.find(L);
00081             if(sit != Letras.end()){ //Si la letra no existe, no hace nada
00082                 L = (*sit);
00083                 L.setFrecuencia(L.getFrecuencia()+1);
00084             }
00085         }
00086     }
00087     return aux;
00088 }
```

```
00089
00090
00091 istream & operator» (istream &is, conjuntoLetras &conjunto){
00092
00093     Letra L;
00094     string primeraLinea;
00095
00096     getline(is, primeraLinea);
00097
00098     if(primeraLinea.empty() || primeraLinea != conjunto.formato){
00099         cerr « "ERROR: No se puede leer el conjunto de letras. La primera linea del "
00100              « "archivo letras.txt debe tener el formato: \n\t" « conjunto.formato « endl;
00101         exit(1);
00102     } else {
00103         while (is » L ){
00104             conjunto.Letras.insert(L);
00105         }
00106     }
00107
00108     return is;
00109 }
00110
00111 ostream & operator« (ostream &os, const conjuntoLetras &conjunto){
00112
00113     os « conjunto.formato « endl;
00114
00115     conjuntoLetras::iterator it;
00116
00117     for(it = conjunto.begin(); it != conjunto.end(); ++it){
00118         os « (*it);
00119     }
00120     return os;
00121
00122 }
00123
00124 conjuntoLetras::iterator conjuntoLetras::begin(){
00125     iterator iterador;
00126     iterador.it = Letras.begin();
00127     return iterador;
00128 }
00129
00130 conjuntoLetras::iterator conjuntoLetras::begin() const{
00131     iterator iterador;
00132     iterador.it = Letras.begin();
00133     return iterador;
00134
00135 }
00136
00137 conjuntoLetras::iterator conjuntoLetras::end(){
00138     iterator iterador;
00139     iterador.it = Letras.end();
00140     return iterador;
00141 }
00142
00143 conjuntoLetras::iterator conjuntoLetras::end() const{
00144     iterator iterador;
00145     iterador.it = Letras.end();
00146     return iterador;
00147 }
00148
00149 conjuntoLetras::iterator & conjuntoLetras::iterator::operator++(){
00150     ++it;
00151     return (*this);
00152 }
00153
00154 bool conjuntoLetras::iterator::operator==(const conjuntoLetras::iterator &i){
00155     return i.it == it;
00156 }
00157
00158 bool conjuntoLetras::iterator::operator!=(const conjuntoLetras::iterator &i){
00159     return i.it != it;
00160 }
00161
00162 Letra conjuntoLetras::iterator::operator* (){
00163     return (*it);
00164 }
```

4.15 src/diccionario.cpp File Reference

```
#include "diccionario.h"
```
Include dependency graph for diccionario.cpp:



Functions

- void intercalado (int c, int intercalar)

  Función externa para mejorar la salida. Hace un salto de linea cada "intercalar" veces.
- ostream & operator<< (ostream &os, const Diccionario &diccionario)
- istream & operator>> (istream &is, Diccionario &d)
- ifstream & operator>> (ifstream &in, Diccionario &d)

4.15.1 Detailed Description

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file diccionario.cpp.

4.15.2 Function Documentation

4.15.2.1 intercalado()

```
void intercalado (
            int c,
            int intercalar )
```

Función externa para mejorar la salida. Hace un salto de linea cada "intercalar" veces.

Parameters

c	Contador con el numero de impresiones hechas
intercalar	Cada cuantas impresiones se quiere dar el salto

Definition at line 13 of file diccionario.cpp.

```
00013                                       {
00014     if(c % intercalar == 0)
00015         cout « endl;
00016 }
```

4.15.2.2 operator<<()

```
ostream & operator<< (
            ostream & os,
            const Diccionario & diccionario )
```

Definition at line 49 of file diccionario.cpp.

```
00049                                                                   {
00050     int c = 1;
00051     for(auto palabra: diccionario){
00052         os « left « setw(25) «  to_string(c) + "-" + palabra;
00053         intercalado(c++,4);
00054     }
00055     cout « endl;
00056
00057     return os;
00058 }
```

4.15.2.3 operator>>() [1/2]

```
ifstream & operator>> (
            ifstream & in,
            Diccionario & d )
```

Definition at line 71 of file diccionario.cpp.

```
00071                                                  { //Archivo de flujo de entrada
00072     string entrada;
00073
00074     while(getline(in, entrada))
00075         d.aniade(entrada);
00076
00077     return in;
00078 }
```

4.15.2.4 operator>>() [2/2]

```
istream & operator>> (
            istream & is,
            Diccionario & d )
```

Definition at line 60 of file diccionario.cpp.

```
00060                                                { //flujo de entrada(ej. cin)
00061     string palabra;
00062
00063     is » palabra;
00064     d.aniade(palabra);
00065
00066     return is;
00067 }
```

4.16 diccionario.cpp

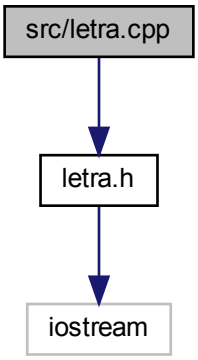
Go to the documentation of this file.

```
00001
00008 #include "diccionario.h"
00009
00010 using namespace std;
00011
00012 //Funcion auxiliar
00013 void intercalado(int c, int intercalar){
00014     if(c % intercalar == 0)
00015         cout « endl;
00016 }
00017
00018 Diccionario::Diccionario(const Diccionario &diccionario) {
00019     *this = diccionario;
00020 }
00021
00022 size_t Diccionario::size() const {
00023     return datos.size();
00024 }
00025
00026 bool Diccionario::Esta(const string & palabra) const {
00027     return datos.find( palabra) != datos.end();
00028 }
00029
00030 vector<string> Diccionario::PalabrasLongitud(const size_t longitud) const{
00031     vector<string> resultado;
00032
00033     for(auto palabra: datos)
00034         if(palabra.size() == longitud)
00035             resultado.push_back(palabra);
00036
00037     return resultado;
00038 }
00039
00040 void Diccionario::aniade(const string &nueva) {
00041     datos.insert( nueva);
00042 }
00043
00044 Diccionario & Diccionario::operator= (const Diccionario &d) {
00045     datos = d.datos;
00046     return *this;
00047 }
00048
00049 ostream & operator« (ostream &os, const Diccionario &diccionario) {
00050     int c = 1;
00051     for(auto palabra: diccionario){
00052         os « left « setw(25) «  to_string(c) + "-" + palabra;
00053         intercalado(c++,4);
00054     }
00055     cout « endl;
00056
00057     return os;
00058 }
00059
00060 istream &operator» (istream &is, Diccionario &d) { //flujo de entrada(ej. cin)
00061     string palabra;
00062
00063     is » palabra;
00064     d.aniade(palabra);
00065
00066     return is;
00067 }
00068
00069
00070
00071 ifstream & operator» (ifstream &in, Diccionario &d ) { //Archivo de flujo de entrada
00072     string entrada;
00073
00074     while(getline(in, entrada))
00075         d.aniade(entrada);
00076
00077     return in;
00078 }
```

4.17 src/letra.cpp File Reference

```
#include "letra.h"
```

Include dependency graph for letra.cpp:



Functions

- istream & operator>> (istream &is, Letra &L)
- ostream & operator<< (ostream &os, const Letra &L)

4.17.1 Detailed Description

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file letra.cpp.

4.17.2 Function Documentation

4.17.2.1 operator<<()

```
ostream & operator<< (
            ostream & os,
            const Letra & L )
```

Parameters

os	el flujo de salida
L	Objeto Letra que se escribe

Returns

el flujo de salida

Definition at line 61 of file letra.cpp.

```
00061                                                {
00062     os « L.caracter « "\t"
00063        « L.frecuencia « "\t"
00064        « L.puntuacion « endl;
00065
00066     return os;
00067 }
```

4.17.2.2 operator>>()

```
istream & operator>> (
             istream & is,
             Letra & L )
```

Parameters

is	el flujo de entrada
L	Objeto Letra donde se realiza la lectura

Returns

el flujo de entrada

Definition at line 53 of file letra.cpp.

```
00053                                               {
00054     is » L.caracter;
00055     is » L.frecuencia;
00056     is » L.puntuacion;
00057
00058     return is;
00059 }
```

4.18 letra.cpp

Go to the documentation of this file.

```
00001
00007 #include "letra.h"
00008
00009 Letra::Letra(){
00010     caracter = 'A'; //Por defecto 'A'
00011     frecuencia = 0;
00012     puntuacion = 0;
00013 }
00014
00015 Letra::Letra(const char &c, const int &frec, const int &punt){
```

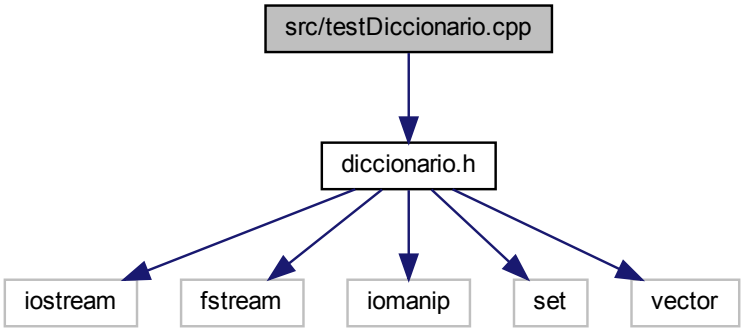
```
00016     caracter = c;
00017     frecuencia = frec;
00018     puntuacion = punt;
00019 }
00020
00021 char Letra::getCaracter() const{
00022     return caracter;
00023 }
00024
00025 int Letra::getPuntuacion() const{
00026    return puntuacion;
00027 }
00028
00029 int Letra::getFrecuencia() const{
00030     return frecuencia;
00031 }
00032
00033 void Letra::setCaracter(const char &c){
00034     caracter = c;
00035 }
00036
00037 void Letra::setFrecuencia( const int &frec){
00038     frecuencia = frec;
00039 }
00040
00041 void Letra::setPuntuacion(const int &punt){
00042     puntuacion = punt;
00043 }
00044
00045 bool Letra::operator== (const Letra &otra) const{
00046     return getCaracter() == otra.getCaracter();
00047 }
00048
00049 bool Letra::operator< (const Letra &otra) const{
00050     return caracter < otra.getCaracter();
00051 }
00052
00053 istream & operator» (istream &is, Letra &L){
00054     is » L.caracter;
00055     is » L.frecuencia;
00056     is » L.puntuacion;
00057
00058     return is;
00059 }
00060
00061 ostream & operator« (ostream &os, const Letra &L){
00062     os « L.caracter « "\t"
00063        « L.frecuencia « "\t"
00064        « L.puntuacion « endl;
00065
00066     return os;
00067 }
```

4.19 src/testDiccionario.cpp File Reference

Archivo de prueba para el diccionario.

```
#include "diccionario.h"
```

Include dependency graph for testDiccionario.cpp:



Functions

- int main (int argc, char ∗argv[ ])

4.19.1 Detailed Description

Archivo de prueba para el diccionario.

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file testDiccionario.cpp.

4.19.2 Function Documentation

4.19.2.1 main()

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

Definition at line 12 of file testDiccionario.cpp.

```
00012                                       {
00013     int c = 1;
00014     if ( argc != 2 ){
00015         cerr  « "Los parámetros son:" « endl
00016               « "\t1.- Fichero con las palabras";
00017
00018         return 0;
00019     }
00020
00022     //EXTRAER EL DICCIONARIO
00024
00025     ifstream f(argv[1]);
00026
00027     if(!f){
00028         cerr « "No se ha podido abrir el fichero " « argv[1] « endl;
00029
00030         return 0;
00031     }
00032
00033     Diccionario D;
00034     cout « "Cargando diccionario..." « endl;
00035
00036     f » D;
00037
00038     cout « "leido el diccionario..." « endl;
00039
00040     cout « D « endl;
00041
00043     //BUSCAR PALABRAS CON X LONGITUD
00045
00046     int longitud;
00047
00048     cout « "Dime la longitud de las palabras que quieres ver: ";
00049
00050     //La longitud no puede ser menor o igual a 0, filtro de entrada
00051     do
00052         cin » longitud;
00053     while (longitud <= 0);
00054
00055     vector<string> v = D.PalabrasLongitud(longitud);
00056
00057     cout « "Hay " « v.size() « " Palabras de Longitud " « longitud « endl;
00058
00059     for(unsigned int i=0; i < v.size(); i++){
00060         cout « left « setw(20) « to_string(c) + "-" + v[i];
00061         intercalado(c++,5);
00062     }
00063     c=1;
00064
00066     //COMPRUEBA PALABRAS
00068
00069     string p;
00070
00071     cout « "\nDime una palabra: ";
00072     cin » p;
00073
00074     if ( D.Esta(p) )
00075         cout « "Sí esa palabra existe\n" « endl;
00076     else
00077         cout « "Esa palabra no existe\n" « endl;
00078
00080     //PROBAR CLASE ITERADORA
00082
00083     cout « "Probando clase iteradora:" « endl;
00084     Diccionario::iterator it;
00085     for(it = D.begin(); it != D.end(); ++it){
00086         cout « left « setw(25) « to_string(c) + "-" + (*it);
00087         intercalado(c++,4);
00088     }
00089     cout « endl;
00090
00091     return 0;
00092 }
```

4.20 testDiccionario.cpp

Go to the documentation of this file.

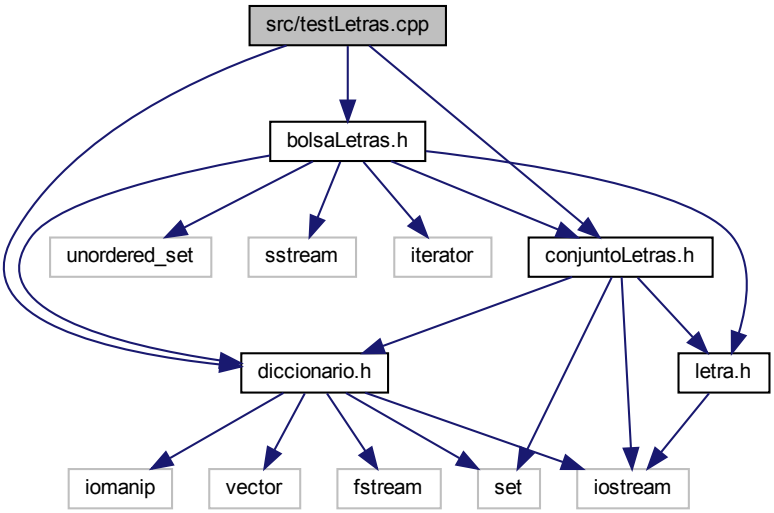
```
00001 #include "diccionario.h"
00002
00003 using namespace std;
00004
00012 int main ( int argc, char * argv[] ){
00013     int c = 1;
00014     if ( argc != 2 ){
00015         cerr  « "Los parámetros son:" « endl
00016               « "\t1.- Fichero con las palabras";
00017
00018         return 0;
00019     }
00020
00022     //EXTRAER EL DICCIONARIO
00024
00025     ifstream f(argv[1]);
00026
00027     if(!f){
00028         cerr « "No se ha podido abrir el fichero " « argv[1] « endl;
00029
00030         return 0;
00031     }
00032
00033     Diccionario D;
00034     cout « "Cargando diccionario..." « endl;
00035
00036     f » D;
00037
00038     cout « "leido el diccionario..." « endl;
00039
00040     cout « D « endl;
00041
00043     //BUSCAR PALABRAS CON X LONGITUD
00045
00046     int longitud;
00047
00048     cout « "Dime la longitud de las palabras que quieres ver: ";
00049
00050     //La longitud no puede ser menor o igual a 0, filtro de entrada
00051     do
00052         cin » longitud;
00053     while (longitud <= 0);
00054
00055     vector<string> v = D.PalabrasLongitud(longitud);
00056
00057     cout « "Hay " « v.size() « " Palabras de Longitud " « longitud « endl;
00058
00059     for(unsigned int i=0; i < v.size(); i++){
00060         cout « left « setw(20) « to_string(c) + "-" + v[i];
00061         intercalado(c++,5);
00062     }
00063     c=1;
00064
00066     //COMPRUEBA PALABRAS
00068
00069     string p;
00070
00071     cout « "\nDime una palabra: ";
00072     cin » p;
00073
00074     if ( D.Esta(p) )
00075         cout « "Sí esa palabra existe\n" « endl;
00076     else
00077         cout « "Esa palabra no existe\n" « endl;
00078
00080     //PROBAR CLASE ITERADORA
00082
00083     cout « "Probando clase iteradora:" « endl;
00084     Diccionario::iterator it;
00085     for(it = D.begin(); it != D.end(); ++it){
00086         cout « left « setw(25) « to_string(c) + "-" + (*it);
00087         intercalado(c++,4);
00088     }
00089     cout « endl;
00090
00091     return 0;
00092 }
```

4.21 src/testLetras.cpp File Reference

Archivo de prueba para el juego de las Letras. Requiere de: -un diccionario.txt -un letras.txt(con el formato dado en el guion) -el numero de letras a generar -modalidad.

```
#include "diccionario.h"
#include "conjuntoLetras.h"
#include "bolsaLetras.h"
```

Include dependency graph for testLetras.cpp:



Functions

- int main (int argc, char ∗argv[ ])

4.21.1 Detailed Description

Archivo de prueba para el juego de las Letras. Requiere de: -un diccionario.txt -un letras.txt(con el formato dado en el guion) -el numero de letras a generar -modalidad.

Author

Yeray López Ramírez

Jaime Castillo Ucles

Date

Diciembre 2021

Definition in file testLetras.cpp.

4.21.2 Function Documentation

4.21.2.1 main()

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

Definition at line 18 of file testLetras.cpp.

```
00018                                   {
00019
00020     if (argc != 5){
00021         cerr « "Numero incorrecto de argumentos." « endl
00022              « "Uso : " « argv[0] « " <Diccionario> <letras> <numero_letras> <modalidad de juego> "
00023              « endl
00024              « "Modos de juego: " « endl « "\t L (Longitud de palabra) "
00025              « "\t P (Puntuacion de la palabra)" « endl;
00026         exit(1);
00027     }
00028
00029     int nLetras = atoi(argv[3]);
00030
00031     if (nLetras <= 0){
00032         cout « "El numero de letras a jugar no puede ser menor a 0" « endl;
00033         exit(2);
00034     }
00035
00036     if (argv[4][0] != 'P' && argv[4][0] != 'L'){
00037         cout « "Modalidades de juego incorrectas" « endl;
00038         exit(3);
00039     }
00040
00041     ifstream f(argv[1]);
00042
00043     if(!f) {
00044         cout « "No se puede abrir el fichero" « argv[1] « endl;
00045         exit(4);
00046     }
00047
00048     Diccionario d;
00049     f » d;
00050
00051     f.close();
00052
00053     conjuntoLetras Letras;
00054     f.open(argv[2], fstream::in);
00055
00056     if(!f) {
00057         cout « "No se puede abrir el fichero" « argv[2] « endl;
00058         exit(5);
00059     }
00060
00061     f » Letras;
00062
00063     f.close();
00064
00065     bolsaLetras bolsa(Letras);
00066
00067     bool salir = false;
00068     char jugar;
00069
00070     cout « "Modo de Juego:";
00071     if (argv[4][0] == 'P')
00072         cout « "PUNTUACIÓN" « endl;
00073     else{
00074         cout « " LONGITUD" « endl;
00075     }
00076
00077     while(!salir){
00078         bolsaLetras nueva(bolsa.getLetras( nLetras )); //Lo llama pocas veces, no pasa nada
00079
00080         cout « "Las letras a jugar son: ";
00081
00082         cout « nueva;
00083
00084         string palabraUsuario;
00085
```

```
00086         cout « endl « "Tu respuesta: ";
00087         cin » palabraUsuario;
00088 
00089         if (nueva.palabraValida(palabraUsuario)){
00090             if(d.Esta(palabraUsuario)){
00091                 cout « "\nPalabra encontrada: " « palabraUsuario « "\t Puntuacion: " «
      Letras.PuntuacionPalabra(palabraUsuario, argv[4][0]) « endl « endl;
00092             }
00093             else{
00094                 cout « endl « "La palabra '"« palabraUsuario « "' no existe" « endl « endl;
00095             }
00096         }
00097         else{
00098             cout « endl « "La palabra dada contiene caracteres que no se han proporcionado" « endl;
00099         }
00100 
00101         set<pair<int,string» soluciones = nueva.getSoluciones(Letras, d, argv[4][0]);
00102 
00103         set<pair<int,string»::const_iterator it;
00104 
00105         cout « "Soluciones posibles: " « endl;
00106         for(it = soluciones.begin(); it != soluciones.end(); ++it){
00107             cout « " Puntuacion: " « it->first « " - Palabra: " « it->second « endl;
00108         }
00109 
00110         if(soluciones.size() > 0){
00111             cout « endl « "La mejor solucion es: '"
00112                  « soluciones.rbegin()->second « "' con una puntuacion de "
00113                  « soluciones.rbegin()->first « " puntos." « endl;
00114 
00115         }else {
00116             cout « endl « "No existen soluciones posibles las letras dadas y en el diccionario " «
      argv[1] « endl;
00117         }
00118 
00119         do{
00120             cout « "¿Quieres seguir jugando? [S/N] ";
00121 
00122             cin » jugar; //convierte a mayuscula
00123             jugar = toupper(jugar);
00124         }while(jugar != 'S' && jugar != 'N');
00125 
00126         if (jugar != 'S')
00127             salir = true;
00128     }
00129 
00130     return 0;
00131 }
```

4.22 testLetras.cpp

Go to the documentation of this file.

```
00001 #include "diccionario.h"
00002 #include "conjuntoLetras.h"
00003 #include "bolsaLetras.h"
00004 
00005 using namespace std;
00006 
00018 int main(int argc, char * argv[]){
00019 
00020     if (argc != 5){
00021         cerr « "Numero incorrecto de argumentos." « endl
00022              « "Uso : " « argv[0] « " <Diccionario> <letras> <numero_letras> <modalidad de juego> "
00023              « endl
00024              « "Modos de juego: " « endl « "\t L (Longitud de palabra) "
00025              « "\t P (Puntuacion de la palabra)" « endl;
00026         exit(1);
00027     }
00028 
00029     int nLetras = atoi(argv[3]);
00030 
00031     if (nLetras <= 0){
00032         cout « "El numero de letras a jugar no puede ser menor a 0" « endl;
00033         exit(2);
00034     }
00035 
00036     if (argv[4][0] != 'P' && argv[4][0] != 'L'){
00037         cout « "Modalidades de juego incorrectas" « endl;
00038         exit(3);
00039     }
00040 
```

```
00041     ifstream f(argv[1]);
00042
00043     if(!f) {
00044         cout « "No se puede abrir el fichero" « argv[1] « endl;
00045         exit(4);
00046     }
00047
00048     Diccionario d;
00049     f » d;
00050
00051     f.close();
00052
00053     conjuntoLetras Letras;
00054     f.open(argv[2], fstream::in);
00055
00056     if(!f) {
00057         cout « "No se puede abrir el fichero" « argv[2] « endl;
00058         exit(5);
00059     }
00060
00061     f » Letras;
00062
00063     f.close();
00064
00065     bolsaLetras bolsa(Letras);
00066
00067     bool salir = false;
00068     char jugar;
00069
00070     cout « "Modo de Juego:";
00071     if (argv[4][0] == 'P')
00072         cout « "PUNTUACIÓN" « endl;
00073     else{
00074         cout « " LONGITUD" « endl;
00075     }
00076
00077     while(!salir){
00078         bolsaLetras nueva(bolsa.getLetras( nLetras )); //Lo llama pocas veces, no pasa nada
00079
00080         cout « "Las letras a jugar son: ";
00081
00082         cout « nueva;
00083
00084         string palabraUsuario;
00085
00086         cout « endl « "Tu respuesta: ";
00087         cin » palabraUsuario;
00088
00089         if (nueva.palabraValida(palabraUsuario)){
00090             if(d.Esta(palabraUsuario)){
00091                 cout « "\nPalabra encontrada: " « palabraUsuario « "\t Puntuacion: " «
    Letras.PuntuacionPalabra(palabraUsuario, argv[4][0]) « endl « endl;
00092             }
00093             else{
00094                 cout « endl « "La palabra '"« palabraUsuario « "' no existe" « endl « endl;
00095             }
00096         }
00097         else{
00098             cout « endl « "La palabra dada contiene caracteres que no se han proporcionado" « endl;
00099         }
00100
00101         set<pair<int,string» soluciones = nueva.getSoluciones(Letras, d, argv[4][0]);
00102
00103         set<pair<int,string»::const_iterator it;
00104
00105         cout « "Soluciones posibles: " « endl;
00106         for(it = soluciones.begin(); it != soluciones.end(); ++it){
00107             cout « " Puntuacion: " « it->first « " - Palabra: " « it->second « endl;
00108         }
00109
00110         if(soluciones.size() > 0){
00111             cout « endl « "La mejor solucion es: '"
00112                  « soluciones.rbegin()->second « "' con una puntuacion de "
00113                  « soluciones.rbegin()->first « " puntos." « endl;
00114
00115         }else {
00116             cout « endl « "No existen soluciones posibles las letras dadas y en el diccionario " «
    argv[1] « endl;
00117         }
00118
00119         do{
00120             cout « "¿Quieres seguir jugando? [S/N] ";
00121
00122             cin » jugar; //convierte a mayuscula
00123             jugar = toupper(jugar);
00124         }while(jugar != 'S' && jugar != 'N');
00125
```

```
00126         if (jugar != 'S')
00127             salir = true;
00128     }
00129 
00130     return 0;
00131 }
```

Index

aniade
  Diccionario, 25
aniadeLetra
  bolsaLetras, 7, 8
  conjuntoLetras, 17

begin
  bolsaLetras, 8
  conjuntoLetras, 17
  Diccionario, 26
bolsaLetras, 5
  aniadeLetra, 7, 8
  begin, 8
  bolsaLetras, 6
  bolsaLetras::iterator, 31
  end, 9
  Esta, 9, 10
  getBolsa, 10
  getLetras, 10
  getSoluciones, 11
  operator<<, 14
  operator>>, 15
  palabraValida, 12
  quitaLetra, 13
  tama, 14
bolsaLetras.cpp
  operator<<, 53
  operator>>, 53
bolsaLetras::iterator, 29
  bolsaLetras, 31
  operator!=, 29
  operator*, 30
  operator++, 30
  operator==, 30

cantidadLetras.cpp
  main, 57
conjuntoLetras, 15
  aniadeLetra, 17
  begin, 17
  conjuntoLetras::iterator, 33
  contarLetras, 17
  end, 18, 19
  Esta, 19
  formato, 24
  frecuenciaTotal, 20
  getLetra, 20, 21
  getLetras, 21
  operator<<, 23
  operator>>, 23
  PuntuacionPalabra, 21
  quitaLetra, 22
  tama, 23
conjuntoLetras.cpp
  operator<<, 61
  operator>>, 62
conjuntoLetras::iterator, 31
  conjuntoLetras, 33
  operator!=, 32
  operator*, 32
  operator++, 32
  operator==, 33
contarLetras
  conjuntoLetras, 17

Diccionario, 24
  aniade, 25
  begin, 26
  Diccionario, 25
  Diccionario::iterator, 35
  end, 26
  Esta, 26
  operator<<, 28
  operator>>, 28
  operator=, 27
  PalabrasLongitud, 27
  size, 27
diccionario.cpp
  intercalado, 65
  operator<<, 66
  operator>>, 66
diccionario.h
  intercalado, 49
Diccionario::iterator, 34
  Diccionario, 35
  operator!=, 34
  operator*, 34
  operator++, 34
  operator==, 35

end
  bolsaLetras, 9
  conjuntoLetras, 18, 19
  Diccionario, 26
Esta
  bolsaLetras, 9, 10
  conjuntoLetras, 19
  Diccionario, 26

formato

conjuntoLetras, 24
frecuenciaTotal
conjuntoLetras, 20

getBolsa
bolsaLetras, 10
getCaracter
Letra, 37
getFrecuencia
Letra, 37
getLetra
conjuntoLetras, 20, 21
getLetras
bolsaLetras, 10
conjuntoLetras, 21
getPuntuacion
Letra, 37
getSoluciones
bolsaLetras, 11

include/bolsaLetras.h, 43, 44
include/conjuntoLetras.h, 45, 46
include/diccionario.h, 47, 49
include/letra.h, 50, 52
intercalado
diccionario.cpp, 65
diccionario.h, 49

Letra, 35
getCaracter, 37
getFrecuencia, 37
getPuntuacion, 37
Letra, 36
operator<, 37
operator<<, 40
operator>>, 40
operator==, 38
setCaracter, 39
setFrecuencia, 39
setPuntuacion, 39
letra.cpp
operator<<, 68
operator>>, 69

main
cantidadLetras.cpp, 57
testDiccionario.cpp, 71
testLetras.cpp, 75

operator!=
bolsaLetras::iterator, 29
conjuntoLetras::iterator, 32
Diccionario::iterator, 34
operator<
Letra, 37
operator<<
bolsaLetras, 14
bolsaLetras.cpp, 53
conjuntoLetras, 23
conjuntoLetras.cpp, 61
Diccionario, 28
diccionario.cpp, 66
Letra, 40
letra.cpp, 68
operator>>
bolsaLetras, 15
bolsaLetras.cpp, 53
conjuntoLetras, 23
conjuntoLetras.cpp, 62
Diccionario, 28
diccionario.cpp, 66
Letra, 40
letra.cpp, 69
operator*
bolsaLetras::iterator, 30
conjuntoLetras::iterator, 32
Diccionario::iterator, 34
operator++
bolsaLetras::iterator, 30
conjuntoLetras::iterator, 32
Diccionario::iterator, 34
operator=
Diccionario, 27
operator==
bolsaLetras::iterator, 30
conjuntoLetras::iterator, 33
Diccionario::iterator, 35
Letra, 38

PalabrasLongitud
Diccionario, 27
palabraValida
bolsaLetras, 12
PuntuacionPalabra
conjuntoLetras, 21

quitaLetra
bolsaLetras, 13
conjuntoLetras, 22

setCaracter
Letra, 39
setFrecuencia
Letra, 39
setPuntuacion
Letra, 39
size
Diccionario, 27
src/bolsaLetras.cpp, 52, 54
src/cantidadLetras.cpp, 56, 59
src/conjuntoLetras.cpp, 61, 63
src/diccionario.cpp, 65, 67
src/letra.cpp, 68, 69
src/testDiccionario.cpp, 70, 73
src/testLetras.cpp, 74, 76

tama
bolsaLetras, 14

conjuntoLetras, 23
testDiccionario.cpp
main, 71
testLetras.cpp
main, 75