

Trabajo Final ED Practicas

Generated by Doxygen 1.9.8

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Dictionary::const_iterator Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 const_iterator() [1/2]	6
3.1.2.2 const_iterator() [2/2]	6
3.1.3 Member Function Documentation	6
3.1.3.1 operator!=(())	6
3.1.3.2 operator*()	6
3.1.3.3 operator++()	6
3.1.3.4 operator==(())	6
3.1.4 Friends And Related Symbol Documentation	6
3.1.4.1 Dictionary	6
3.1.5 Member Data Documentation	7
3.1.5.1 it	7
3.2 LetterSet::const_iterator Class Reference	7
3.2.1 Detailed Description	7
3.2.2 Constructor & Destructor Documentation	7
3.2.2.1 const_iterator() [1/2]	7
3.2.2.2 const_iterator() [2/2]	8
3.2.3 Member Function Documentation	8
3.2.3.1 operator!=(())	8
3.2.3.2 operator*()	8
3.2.3.3 operator++()	8
3.2.3.4 operator->()	8
3.2.3.5 operator==(())	8
3.2.4 Friends And Related Symbol Documentation	8
3.2.4.1 LetterSet	8
3.2.5 Member Data Documentation	8
3.2.5.1 it	8
3.3 Dictionary Class Reference	9
3.3.1 Constructor & Destructor Documentation	10
3.3.1.1 Dictionary()	10
3.3.1.2 ~Dictionary()	10
3.3.2 Member Function Documentation	10
3.3.2.1 begin() [1/2]	10
3.3.2.2 begin() [2/2]	10

3.3.2.3 clear()	10
3.3.2.4 empty()	11
3.3.2.5 end() [1/2]	11
3.3.2.6 end() [2/2]	11
3.3.2.7 erase()	11
3.3.2.8 exists()	11
3.3.2.9 find()	12
3.3.2.10 getOccurrences()	12
3.3.2.11 getTotalLetters()	12
3.3.2.12 getWordsLength()	12
3.3.2.13 insert()	13
3.3.2.14 range_prefix()	13
3.3.2.15 size()	13
3.3.3 Friends And Related Symbol Documentation	13
3.3.3.1 operator<<	13
3.3.3.2 operator>>	14
3.3.4 Member Data Documentation	14
3.3.4.1 words	14
3.4 Dictionary::iterator Class Reference	15
3.4.1 Detailed Description	15
3.4.2 Constructor & Destructor Documentation	15
3.4.2.1 iterator() [1/2]	15
3.4.2.2 iterator() [2/2]	15
3.4.3 Member Function Documentation	16
3.4.3.1 operator!=(())	16
3.4.3.2 operator*()	16
3.4.3.3 operator++()	16
3.4.3.4 operator==(())	16
3.4.4 Friends And Related Symbol Documentation	16
3.4.4.1 Dictionary	16
3.4.5 Member Data Documentation	16
3.4.5.1 it	16
3.5 LettersBag::iterator Class Reference	16
3.5.1 Detailed Description	17
3.5.2 Constructor & Destructor Documentation	17
3.5.2.1 iterator() [1/2]	17
3.5.2.2 iterator() [2/2]	17
3.5.3 Member Function Documentation	17
3.5.3.1 operator!=(())	17
3.5.3.2 operator*()	17
3.5.3.3 operator++()	18
3.5.3.4 operator==(())	18

3.5.4 Member Data Documentation	18
3.5.4.1 it	18
3.5.4.2 LettersBag	18
3.6 LetterSet::iterator Class Reference	18
3.6.1 Detailed Description	19
3.6.2 Constructor & Destructor Documentation	19
3.6.2.1 iterator() [1/2]	19
3.6.2.2 iterator() [2/2]	19
3.6.3 Member Function Documentation	19
3.6.3.1 operator!=(())	19
3.6.3.2 operator*()	19
3.6.3.3 operator++()	19
3.6.3.4 operator->()	19
3.6.3.5 operator==(())	19
3.6.4 Friends And Related Symbol Documentation	20
3.6.4.1 LetterSet	20
3.6.5 Member Data Documentation	20
3.6.5.1 it	20
3.7 LetterInfo Struct Reference	20
3.7.1 Detailed Description	20
3.7.2 Constructor & Destructor Documentation	20
3.7.2.1 LetterInfo() [1/2]	20
3.7.2.2 LetterInfo() [2/2]	20
3.7.3 Member Data Documentation	21
3.7.3.1 repetitions	21
3.7.3.2 score	21
3.8 LettersBag Class Reference	21
3.8.1 Detailed Description	22
3.8.2 Constructor & Destructor Documentation	22
3.8.2.1 LettersBag() [1/2]	22
3.8.2.2 LettersBag() [2/2]	22
3.8.3 Member Function Documentation	22
3.8.3.1 begin()	22
3.8.3.2 clear()	23
3.8.3.3 empty()	23
3.8.3.4 end()	23
3.8.3.5 erase()	23
3.8.3.6 getLetter()	23
3.8.3.7 size()	24
3.8.3.8 toString()	24
3.8.4 Member Data Documentation	24
3.8.4.1 bag	24

3.9 LetterSet Class Reference	24
3.9.1 Detailed Description	25
3.9.2 Constructor & Destructor Documentation	25
3.9.2.1 LetterSet()	25
3.9.3 Member Function Documentation	25
3.9.3.1 begin() [1/2]	25
3.9.3.2 begin() [2/2]	26
3.9.3.3 empty()	26
3.9.3.4 end() [1/2]	26
3.9.3.5 end() [2/2]	26
3.9.3.6 getLetterInfo()	26
3.9.3.7 size()	26
3.9.4 Member Data Documentation	27
3.9.4.1 charSet	27
3.10 solucion Struct Reference	27
3.10.1 Detailed Description	27
3.10.2 Member Data Documentation	27
3.10.2.1 cantidad	27
3.10.2.2 op	27
3.10.2.3 valor	28
3.11 Solver Class Reference	28
3.11.1 Detailed Description	28
3.11.2 Constructor & Destructor Documentation	28
3.11.2.1 Solver()	28
3.11.3 Member Function Documentation	29
3.11.3.1 existe()	29
3.11.3.2 getSolutions()	29
3.11.3.3 poderConstruir()	29
3.11.3.4 puntosPalabra()	31
3.11.4 Member Data Documentation	31
3.11.4.1 dictionary	31
3.11.4.2 ls	31
4 File Documentation	33
4.1 include/dictionary.h File Reference	33
4.2 dictionary.h	33
4.3 include/letters_bag.h File Reference	34
4.4 letters_bag.h	35
4.5 include/letters_set.h File Reference	36
4.6 letters_set.h	36
4.7 include/solver.h File Reference	37
4.8 solver.h	37

4.9 src/cantidad_letras.cpp File Reference	38
4.10 src/cifras.cpp File Reference	38
4.10.1 Function Documentation	39
4.10.1.1 Cifras()	39
4.10.1.2 GeneraOperaciones()	39
4.10.1.3 main()	39
4.11 src/dictionary.cpp File Reference	40
4.11.1 Function Documentation	40
4.11.1.1 operator<<()	40
4.11.1.2 operator>>()	40
4.12 src/letras.cpp File Reference	41
4.12.1 Function Documentation	41
4.12.1.1 main()	41
4.12.1.2 ModoPalabraMasLarga()	41
4.12.1.3 ModoPalabraMayorPuntuacion()	41
4.13 src/letters_bag.cpp File Reference	42
4.14 src/letters_set.cpp File Reference	42
4.15 src/solver.cpp File Reference	42
4.16 src/testdiccionario.cpp File Reference	42
4.16.1 Function Documentation	42
4.16.1.1 main()	42
Index	43

Chapter 1

Class Index

1.1 Class List

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

Dictionary::const_iterator	
Iterador constante del diccionario. Obtiene las palabras ordenadas alfabéticamente	5
LetterSet::const_iterator	
Iterador constante para recorrer el conjunto de letras	7
Dictionary	9
Dictionary::iterator	
Iterador del diccionario. Obtiene las palabras ordenadas alfabéticamente	15
LettersBag::iterator	
Iterador para recorrer la bolsa de letras	16
LetterSet::iterator	
Iterador para recorrer el conjunto de letras	18
LetterInfo	
Estructura para almacenar información sobre una letra	20
LettersBag	
TDA LettersBag	21
LetterSet	
TDA LetterSet	24
solucion	
Cifras Utilizamos dos funciones y un struct para calcular las operaciones necesarias para llegar a un número aleatorio de 3 cifras generado en el main, utilizando los números de un que se dan	27
Solver	
TDA Solver	28

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

include/dictionary.h	33
include/letters_bag.h	34
include/letters_set.h	36
include/solver.h	37
src/cantidad_letras.cpp	38
src/cifras.cpp	38
src/dictionary.cpp	40
src/letras.cpp	41
src/letters_bag.cpp	42
src/letters_set.cpp	42
src/solver.cpp	42
src/testdiccionario.cpp	42

Chapter 3

Class Documentation

3.1 Dictionary::const_iterator Class Reference

Iterador constante del diccionario. Obtiene las palabras ordenadas alfabéticamente.

```
#include <dictionary.h>
```

Collaboration diagram for Dictionary::const_iterator:

Public Member Functions

- [const_iterator](#) ()
Iterador constante del diccionario. Obtiene las palabras ordenadas alfabéticamente.
- [const_iterator](#) (set< string >::const_iterator)
- string [operator*](#) ()
- [const_iterator](#) & [operator++](#) ()
- bool [operator==](#) (const [const_iterator](#) &i)
- bool [operator!=](#) (const [const_iterator](#) &i)

Private Attributes

- set< string >::[const_iterator](#) it

Friends

- class [Dictionary](#)

3.1.1 Detailed Description

Iterador constante del diccionario. Obtiene las palabras ordenadas alfabéticamente.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 `const_iterator()` [1/2]

```
Dictionary::const_iterator::const_iterator ( )
```

Iterador constante del diccionario. Obtiene las palabras ordenadas alfabéticamente.

3.1.2.2 `const_iterator()` [2/2]

```
Dictionary::const_iterator::const_iterator (
    set< string >::const_iterator otro )
```

3.1.3 Member Function Documentation

3.1.3.1 `operator!=(())`

```
bool Dictionary::const_iterator::operator!= (
    const const_iterator & i )
```

3.1.3.2 `operator*()`

```
string Dictionary::const_iterator::operator* ( )
```

3.1.3.3 `operator++()`

```
Dictionary::const_iterator & Dictionary::const_iterator::operator++ ( )
```

3.1.3.4 `operator==(())`

```
bool Dictionary::const_iterator::operator== (
    const const_iterator & i )
```

3.1.4 Friends And Related Symbol Documentation

3.1.4.1 Dictionary

```
friend class Dictionary [friend]
```

3.1.5 Member Data Documentation

3.1.5.1 it

```
set<string>::const_iterator Dictionary::const_iterator::it [private]
```

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

- include/dictionary.h
- src/dictionary.cpp

3.2 LetterSet::const_iterator Class Reference

Iterador constante para recorrer el conjunto de letras.

```
#include <letters_set.h>
```

Collaboration diagram for LetterSet::const_iterator:

Public Member Functions

- [const_iterator](#) ()
- [const_iterator](#) (map< char, [LetterInfo](#) >::const_iterator)
- const pair< const char, [LetterInfo](#) > & [operator*](#) ()
- const pair< const char, [LetterInfo](#) > * [operator->](#) ()
- [const_iterator](#) & [operator++](#) ()
- bool [operator!=](#) (const [const_iterator](#) &) const
- bool [operator==](#) (const [const_iterator](#) &) const

Private Attributes

- map< char, [LetterInfo](#) >::const_iterator it

Friends

- class [LetterSet](#)

3.2.1 Detailed Description

Iterador constante para recorrer el conjunto de letras.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 const_iterator() [1/2]

```
LetterSet::const_iterator::const_iterator ( )
```

3.2.2.2 const_iterator() [2/2]

```
LetterSet::const_iterator::const_iterator (
    map< char, LetterInfo >::const_iterator iter )
```

3.2.3 Member Function Documentation

3.2.3.1 operator!=(())

```
bool LetterSet::const_iterator::operator!= (
    const const_iterator & other ) const
```

3.2.3.2 operator*()

```
const pair< const char, LetterInfo > & LetterSet::const_iterator::operator* ( )
```

3.2.3.3 operator++()

```
LetterSet::const_iterator & LetterSet::const_iterator::operator++ ( )
```

3.2.3.4 operator->()

```
const pair< const char, LetterInfo > * LetterSet::const_iterator::operator-> ( )
```

3.2.3.5 operator==(())

```
bool LetterSet::const_iterator::operator== (
    const const_iterator & other ) const
```

3.2.4 Friends And Related Symbol Documentation

3.2.4.1 LetterSet

```
friend class LetterSet [friend]
```

3.2.5 Member Data Documentation

3.2.5.1 it

```
map<char,LetterInfo>::const_iterator LetterSet::const_iterator::it [private]
```

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

- include/letters_set.h
- src/letters_set.cpp

3.3 Dictionary Class Reference

```
#include <dictionary.h>
```

Classes

- class `const_iterator`
Iterador constante del diccionario. Obtiene las palabras ordenadas alfabéticamente.
- class `iterator`
Iterador del diccionario. Obtiene las palabras ordenadas alfabéticamente.

Public Member Functions

- `Dictionary ()`
Constructor por defecto.
- `~Dictionary ()`
Destructor.
- `void clear ()`
Limpia el `Dictionary`.
- `unsigned int size () const`
Tamaño del diccionario.
- `bool empty () const`
Comprueba si el diccionario está vacío.
- `bool exists (const string &val)`
Indica si una palabra está en el diccionario o no. Este método comprueba si una determinada palabra se encuentra o no en el diccionario.
- `bool erase (const string &val)`
Elimina una palabra del diccionario.
- `int getOccurrences (const char c) const`
Indica el número de apariciones de una letra.
- `int getTotalLetters () const`
Indica el número total de letras.
- `iterator find (const string &w)`
Indica si una palabra está en el diccionario.
- `vector< string > getWordsLength (int longitud)`
Obtiene todas las palabras de una longitud param longitud: valor de longitud de la palabras a devolver.
- `pair< iterator, bool > insert (const string &val)`
Inserta una palabra en el diccionario.
- `pair< iterator, iterator > range_prefix (const string &val)`
Busca un las palabras con un prefijo.
- `iterator begin ()`
Obtiene el iterador apuntando a la primera palabra del diccionario.
- `const_iterator begin () const`
- `iterator end ()`
- `const_iterator end () const`
Obtiene el iterador apuntando al final del diccionario.

Private Attributes

- `set< string > words`

Friends

- `istream & operator>>` (`istream &is`, [Dictionary](#) &dic)
Sobrecarga del operador de entrada.
- `ostream & operator<<` (`ostream &os`, const [Dictionary](#) &dic)
Sobrecarga del operador de salida.

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Dictionary()

```
Dictionary::Dictionary ( )
```

Constructor por defecto.

Crea un [Dictionary](#) vacío

3.3.1.2 ~Dictionary()

```
Dictionary::~~Dictionary ( )
```

Destructor.

3.3.2 Member Function Documentation

3.3.2.1 begin() [1/2]

```
Dictionary::iterator Dictionary::begin ( )
```

Obtiene el iterador apuntando a la primera palabra del diccionario.

3.3.2.2 begin() [2/2]

```
Dictionary::const_iterator Dictionary::begin ( ) const
```

3.3.2.3 clear()

```
void Dictionary::clear ( )
```

Limpia el [Dictionary](#).

Postcondition

el diccionario queda con 0 palabras

3.3.2.4 empty()

```
bool Dictionary::empty ( ) const
```

Comprueba si el diccionario está vacío.

Returns

true si el diccionario está vacío, false en caso contrario

3.3.2.5 end() [1/2]

```
Dictionary::iterator Dictionary::end ( )
```

Obtiene el iterador apuntando al final del diccionario.

3.3.2.6 end() [2/2]

```
Dictionary::const_iterator Dictionary::end ( ) const
```

3.3.2.7 erase()

```
bool Dictionary::erase (
    const string & val )
```

Elimina una palabra del diccionario.

Parameters

<i>val</i>	Palabra a borrar del diccionario
------------	----------------------------------

Returns

Booleano que indica si la palabra se ha borrado del diccionario

3.3.2.8 exists()

```
bool Dictionary::exists (
    const string & val )
```

Indica si una palabra esta en el diccionario o no. Este método comprueba si una determinada palabra se encuentra o no en el diccionario.

Parameters

<i>palabra</i>	la palabra que se quiere buscar.
----------------	----------------------------------

Returns

Booleano indicando si la palabra existe o no en el diccionario

3.3.2.9 find()

```
Dictionary::iterator Dictionary::find (
    const string & w )
```

Indica si una palabra esta en el diccionario.

Returns

iterador apuntando a la palabra si esta o end si no esta

3.3.2.10 getOccurrences()

```
int Dictionary::getOccurrences (
    const char c ) const
```

Indica el numero de apariciones de una letra.

Parameters

<i>c</i>	letra a buscar.
----------	-----------------

Returns

Un entero indicando el numero de apariciones.

3.3.2.11 getTotalLetters()

```
int Dictionary::getTotalLetters ( ) const
```

Indica el numero totales de letras.

Returns

Un entero indicando el numero totales de letras

3.3.2.12 getWordsLength()

```
vector< string > Dictionary::getWordsLength (
    int longitud )
```

Obtiene todas las palabras de una longitud param longitud: valor de longitud de la palabras a devolver.

Returns

un vector con palabra de la longitud dada

3.3.2.13 insert()

```
std::pair< Dictionary::iterator, bool > Dictionary::insert (
    const string & val )
```

Inserta una palabra en el diccionario.

Parameters

<i>val</i>	palabra a insertar en el diccionario
------------	--------------------------------------

Returns

Booleano que indica si la inserción ha tenido éxito. Una palabra se inserta con éxito si no existía previamente en el diccionario. El iterador apunta a la palabra

3.3.2.14 range_prefix()

```
std::pair< Dictionary::iterator, Dictionary::iterator > Dictionary::range_prefix (
    const string & val )
```

Busca un las palabras con un prefijo.

Parameters

<i>val</i>	prefijo a buscar
------------	------------------

Returns

un pair con dos iteradores el primero apuntando a la primera palabra con el prefijo y el segundo donde ya no contiene el prefijo. Si no existe el prefijo se devuelve los dos iteradores apuntando a [end\(\)](#)

3.3.2.15 size()

```
unsigned int Dictionary::size ( ) const
```

Tamaño del diccionario.

Returns

Número de palabras guardadas en el diccionario

3.3.3 Friends And Related Symbol Documentation

3.3.3.1 operator<<

```
ostream & operator<< (
    ostream & os,
    const Dictionary & dic ) [friend]
```

Sobrecarga del operador de salida.

Permite imprimir el diccionario completo a un flujo de salida

Parameters

<i>os</i>	Flujo de salida, donde imprimir el diccionario
<i>dic</i>	Diccionario a imprimir

Returns

Flujo de salida, para poder encadenar el operador

3.3.3.2 operator>>

```
istream & operator>> (
    istream & is,
    Dictionary & dic ) [friend]
```

Sobrecarga del operador de entrada.

Permite leer las palabras de un fichero de texto e introducirlas en el diccionario

Parameters

<i>is</i>	Flujo de entrada
<i>dic</i>	Diccionario a rellenar

Returns

Flujo de entrada para poder encadenar el operador

Permite leer las palabras de un fichero de texto e introducirlas en el diccionario

Parameters

<i>isz</i>	Flujo de entrada
<i>dic</i>	Diccionario a rellenar

Returns

Flujo de entrada para poder encadenar el operador

3.3.4 Member Data Documentation**3.3.4.1 words**

```
set<string> Dictionary::words [private]
```

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

- include/[dictionary.h](#)
- src/[dictionary.cpp](#)

3.4 Dictionary::iterator Class Reference

Iterador del diccionario. Obtiene las palabras ordenadas alfabéticamente.

```
#include <dictionary.h>
```

Collaboration diagram for Dictionary::iterator:

Public Member Functions

- [iterator](#) ()
Iterador del diccionario. Obtiene las palabras ordenadas alfabéticamente.
- [iterator](#) (set< string >::const_iterator)
- string [operator*](#) ()
- [iterator](#) & [operator++](#) ()
- bool [operator==](#) (const [iterator](#) &i)
- bool [operator!=](#) (const [iterator](#) &i)

Private Attributes

- set< string >::iterator [it](#)

Friends

- class [Dictionary](#)

3.4.1 Detailed Description

Iterador del diccionario. Obtiene las palabras ordenadas alfabéticamente.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 iterator() [1/2]

```
Dictionary::iterator::iterator ( )
```

Iterador del diccionario. Obtiene las palabras ordenadas alfabéticamente.

3.4.2.2 iterator() [2/2]

```
Dictionary::iterator::iterator (
    set< string >::const_iterator )
```

3.4.3 Member Function Documentation

3.4.3.1 operator!=(())

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

3.4.3.2 operator*()

```
string Dictionary::iterator::operator* ( )
```

3.4.3.3 operator++()

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

3.4.3.4 operator==(())

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

3.4.4 Friends And Related Symbol Documentation

3.4.4.1 Dictionary

```
friend class Dictionary [friend]
```

3.4.5 Member Data Documentation

3.4.5.1 it

```
set<string>::iterator Dictionary::iterator::it [private]
```

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

- include/[dictionary.h](#)
- src/[dictionary.cpp](#)

3.5 LettersBag::iterator Class Reference

Iterador para recorrer la bolsa de letras.

```
#include <letters_bag.h>
```

Collaboration diagram for LettersBag::iterator:

Public Member Functions

- [iterator](#) ()
- [iterator](#) (vector< char >::iterator)
- char & [operator*](#) ()
- [iterator](#) & [operator++](#) ()
- bool [operator!=](#) (const [iterator](#) &) const
- bool [operator==](#) (const [iterator](#) &) const

Public Attributes

- friend [LettersBag](#)

Private Attributes

- vector< char >::iterator [it](#)

3.5.1 Detailed Description

Iterador para recorrer la bolsa de letras.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 [iterator\(\)](#) [1/2]

```
LettersBag::iterator::iterator ( )
```

3.5.2.2 [iterator\(\)](#) [2/2]

```
LettersBag::iterator::iterator (
    vector< char >::iterator iter )
```

3.5.3 Member Function Documentation

3.5.3.1 [operator"!=\(\)](#)

```
bool LettersBag::iterator::operator!= (
    const iterator & other ) const
```

3.5.3.2 [operator*\(\)](#)

```
char & LettersBag::iterator::operator* ( )
```

3.5.3.3 operator++()

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

3.5.3.4 operator==()

```
bool LettersBag::iterator::operator== (
    const iterator & other ) const
```

3.5.4 Member Data Documentation

3.5.4.1 it

```
vector<char>::iterator LettersBag::iterator::it [private]
```

3.5.4.2 LettersBag

```
friend LettersBag::iterator::LettersBag
```

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

- include/letters_bag.h
- src/letters_bag.cpp

3.6 LetterSet::iterator Class Reference

Iterador para recorrer el conjunto de letras.

```
#include <letters_set.h>
```

Collaboration diagram for LetterSet::iterator:

Public Member Functions

- iterator ()
- iterator (map< char, LetterInfo >::iterator)
- pair< const char, LetterInfo > & operator* ()
- pair< const char, LetterInfo > * operator-> ()
- iterator & operator++ ()
- bool operator!= (const iterator &) const
- bool operator== (const iterator &) const

Private Attributes

- map< char, LetterInfo >::iterator it

Friends

- class [LetterSet](#)

3.6.1 Detailed Description

Iterador para recorrer el conjunto de letras.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 iterator() [1/2]

```
LetterSet::iterator::iterator ( )
```

3.6.2.2 iterator() [2/2]

```
LetterSet::iterator::iterator (
    map< char, LetterInfo >::iterator iter )
```

3.6.3 Member Function Documentation

3.6.3.1 operator!=(())

```
bool LetterSet::iterator::operator!= (
    const iterator & other ) const
```

3.6.3.2 operator*()

```
pair< const char, LetterInfo > & LetterSet::iterator::operator* ( )
```

3.6.3.3 operator++()

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

3.6.3.4 operator->()

```
pair< const char, LetterInfo > * LetterSet::iterator::operator-> ( )
```

3.6.3.5 operator==(())

```
bool LetterSet::iterator::operator== (
    const iterator & other ) const
```

3.6.4 Friends And Related Symbol Documentation

3.6.4.1 LetterSet

```
friend class LetterSet [friend]
```

3.6.5 Member Data Documentation

3.6.5.1 it

```
map<char, LetterInfo>::iterator LetterSet::iterator::it [private]
```

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

- include/letters_set.h
- src/letters_set.cpp

3.7 LetterInfo Struct Reference

Estructura para almacenar información sobre una letra.

```
#include <letters_set.h>
```

Public Member Functions

- [LetterInfo](#) ()
Constructor por defecto.
- [LetterInfo](#) (unsigned int reps, unsigned int [score](#))
Constructor con parámetros.

Public Attributes

- unsigned int [repetitions](#)
- unsigned int [score](#)

3.7.1 Detailed Description

Estructura para almacenar información sobre una letra.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 LetterInfo() [1/2]

```
LetterInfo::LetterInfo ( ) [inline]
```

Constructor por defecto.

3.7.2.2 LetterInfo() [2/2]

```
LetterInfo::LetterInfo (
    unsigned int reps,
    unsigned int score ) [inline]
```

Constructor con parámetros.

Parameters

<i>reps</i>	Número de repeticiones del carácter en la partida
<i>score</i>	Puntuación del carácter

3.7.3 Member Data Documentation

3.7.3.1 repetitions

```
unsigned int LetterInfo::repetitions
```

3.7.3.2 score

```
unsigned int LetterInfo::score
```

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

- [include/letters_set.h](#)

3.8 LettersBag Class Reference

TDA [LettersBag](#).

```
#include <letters_bag.h>
```

Classes

- class [iterator](#)
Iterador para recorrer la bolsa de letras.

Public Member Functions

- [LettersBag](#) (const [LetterSet](#) &)
Constructor de la clase [LettersBag](#) a partir de un [LetterSet](#).
- [LettersBag](#) (const string &)
Constructor de la clase [LettersBag](#) a partir de un string.
- const char [getLetter](#) (int) const
Obtiene una letra de la bolsa en una posición específica.
- const string [toString](#) () const
Convierte el contenido de la bolsa a un string.
- unsigned [size](#) () const
Obtiene el tamaño de la bolsa.
- bool [empty](#) () const
Comprueba si la bolsa está vacía.
- void [clear](#) ()
Limpia el contenido de la bolsa.
- void [erase](#) (const char)
Elimina una ocurrencia de una letra específica de la bolsa.
- [iterator](#) [begin](#) ()
Devuelve un iterador al inicio de la bolsa.
- [iterator](#) [end](#) ()
Devuelve un iterador al final de la bolsa.

Private Attributes

- `vector< char >` [bag](#)

3.8.1 Detailed Description

TDA [LettersBag](#).

Esta clase representa una bolsa de letras que permite la extracción aleatoria de caracteres, útil para generar manos de juego o secuencias aleatorias.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 [LettersBag\(\)](#) [1/2]

```
LettersBag::LettersBag (  
    const LetterSet & conjunto )
```

Constructor de la clase [LettersBag](#) a partir de un [LetterSet](#).

Rellena la bolsa con las letras definidas en el conjunto de letras proporcionado.

Parameters

<i>letterSet</i>	Conjunto de información de letras (LetterSet)
------------------	---

3.8.2.2 [LettersBag\(\)](#) [2/2]

```
LettersBag::LettersBag (  
    const string & nombre )
```

Constructor de la clase [LettersBag](#) a partir de un string.

Parameters

<i>str</i>	Cadena de texto para inicializar la bolsa
------------	---

3.8.3 Member Function Documentation

3.8.3.1 [begin\(\)](#)

```
LettersBag::iterator LettersBag::begin ( )
```

Devuelve un iterador al inicio de la bolsa.

3.8.3.2 clear()

```
void LettersBag::clear ( )
```

Limpia el contenido de la bolsa.

3.8.3.3 empty()

```
bool LettersBag::empty ( ) const
```

Comprueba si la bolsa está vacía.

Returns

true si está vacía, false en caso contrario

3.8.3.4 end()

```
LettersBag::iterator LettersBag::end ( )
```

Devuelve un iterador al final de la bolsa.

3.8.3.5 erase()

```
void LettersBag::erase (
    const char c )
```

Elimina una ocurrencia de una letra específica de la bolsa.

Parameters

<i>c</i>	Carácter a eliminar
----------	---------------------

3.8.3.6 getLetter()

```
const char LettersBag::getLetter (
    int index ) const
```

Obtiene una letra de la bolsa en una posición específica.

Parameters

<i>index</i>	Índice de la letra a recuperar
--------------	--------------------------------

Returns

El carácter en la posición indicada

3.8.3.7 size()

```
unsigned LettersBag::size ( ) const
```

Obtiene el tamaño de la bolsa.

Returns

Número de letras en la bolsa

3.8.3.8 toString()

```
const string LettersBag::toString ( ) const
```

Convierte el contenido de la bolsa a un string.

Returns

Cadena con todas las letras de la bolsa

3.8.4 Member Data Documentation**3.8.4.1 bag**

```
vector<char> LettersBag::bag [private]
```

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

- [include/letters_bag.h](#)
- [src/letters_bag.cpp](#)

3.9 LetterSet Class Reference

TDA [LetterSet](#).

```
#include <letters_set.h>
```

Classes

- class [const_iterator](#)
Iterador constante para recorrer el conjunto de letras.
- class [iterator](#)
Iterador para recorrer el conjunto de letras.

Public Member Functions

- [LetterSet](#) (const string &)
- const [LetterInfo](#) [getLetterInfo](#) (char) const
Getter de la información de la letra.
- bool [empty](#) () const
Comprueba si el conjunto está vacío.
- unsigned [size](#) () const
Obtiene el tamaño del conjunto (número de tipos de letras distintas)
- [iterator](#) [begin](#) ()
Devuelve un iterador al inicio del conjunto.
- [iterator](#) [end](#) ()
Devuelve un iterador al final del conjunto.
- const [_iterator](#) [begin](#) () const
Devuelve un iterador constante al inicio del conjunto.
- const [_iterator](#) [end](#) () const
Devuelve un iterador constante al final del conjunto.

Private Attributes

- map< char, [LetterInfo](#) > [charSet](#)

3.9.1 Detailed Description

TDA [LetterSet](#).

Esta clase gestiona un conjunto de letras, asociando a cada carácter su información de juego (repeticiones permitidas y puntuación).

3.9.2 Constructor & Destructor Documentation

3.9.2.1 LetterSet()

```
LetterSet::LetterSet (  
    const string & nombre )
```

3.9.3 Member Function Documentation

3.9.3.1 begin() [1/2]

```
LetterSet::iterator LetterSet::begin ( )
```

Devuelve un iterador al inicio del conjunto.

3.9.3.2 begin() [2/2]

```
LetterSet::const_iterator LetterSet::begin ( ) const
```

Devuelve un iterador constante al inicio del conjunto.

3.9.3.3 empty()

```
bool LetterSet::empty ( ) const
```

Comprueba si el conjunto está vacío.

Returns

true si no hay letras, false en caso contrario

3.9.3.4 end() [1/2]

```
LetterSet::iterator LetterSet::end ( )
```

Devuelve un iterador al final del conjunto.

3.9.3.5 end() [2/2]

```
LetterSet::const_iterator LetterSet::end ( ) const
```

Devuelve un iterador constante al final del conjunto.

3.9.3.6 getLetterInfo()

```
const LetterInfo LetterSet::getLetterInfo (
    char letter ) const
```

Getter de la información de la letra.

Parameters

<i>val</i>	Carácter del cual se quiere obtener la información
------------	--

Returns

Devuelve la informacion ([LetterInfo](#)) de la letra en el conjunto

3.9.3.7 size()

```
unsigned LetterSet::size ( ) const
```

Obtiene el tamaño del conjunto (número de tipos de letras distintas)

Returns

Entero con el tamaño

3.9.4 Member Data Documentation

3.9.4.1 charSet

```
map<char, LetterInfo> LetterSet::charSet [private]
```

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

- include/letters_set.h
- src/letters_set.cpp

3.10 solucion Struct Reference

Cifras Utilizamos dos funciones y un struct para calcular las operaciones necesarias para llegar a un número aleatorio de 3 cifras generado en el main, utilizando los números de un que se dan.

Public Attributes

- int [valor](#)
- string [op](#)
- int [cantidad](#)

3.10.1 Detailed Description

Cifras Utilizamos dos funciones y un struct para calcular las operaciones necesarias para llegar a un número aleatorio de 3 cifras generado en el main, utilizando los números de un que se dan.

struct solucion Guarda una solución, guardando el valor al que se ha llegado, un string con las operaciones utilizadas para mostrarlas por pantalla y la cantidad de operaciones que han sido necesarias

3.10.2 Member Data Documentation

3.10.2.1 cantidad

```
int solucion::cantidad
```

3.10.2.2 op

```
string solucion::op
```

3.10.2.3 valor

```
int solucion::valor
```

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

- [src/cifras.cpp](#)

3.11 Solver Class Reference

TDA [Solver](#).

```
#include <solver.h>
```

Collaboration diagram for Solver:

Public Member Functions

- [Solver](#) (const [Dictionary](#) &dic, const [LetterSet](#) &letter)
Constructor de la clase [Solver](#).
- vector< string > [getSolutions](#) (const vector< char > &available_letters, bool score_game)
Obtiene las soluciones posibles para una partida.
- int [puntosPalabra](#) (string palabra)
Calcula la puntuación de una palabra.
- bool [poderConstruir](#) (string palabra, vector< char > available_letters)
Comprueba si una palabra se puede construir con un conjunto de letras.
- bool [existe](#) (string palabra)
Comprueba si una palabra existe en el diccionario del [Solver](#).

Private Attributes

- [Dictionary](#) dictionary
- [LetterSet](#) ls

3.11.1 Detailed Description

TDA [Solver](#).

Esta clase se encarga de resolver el juego, encontrando soluciones válidas en el diccionario a partir de un conjunto de letras disponibles.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 Solver()

```
Solver::Solver (
    const Dictionary & dic,
    const LetterSet & letter ) [inline]
```

Constructor de la clase [Solver](#).

Parameters

<i>dic</i>	Diccionario con las palabras válidas
<i>letter</i>	Conjunto de letras con sus puntuaciones

3.11.3 Member Function Documentation

3.11.3.1 existe()

```
bool Solver::existe (
    string palabra )
```

Comprueba si una palabra existe en el diccionario del [Solver](#).

Parameters

<i>palabra</i>	Palabra a buscar
----------------	------------------

Returns

true si existe, false si no

3.11.3.2 getSolutions()

```
vector< string > Solver::getSolutions (
    const vector< char > & available_letters,
    bool score_game )
```

Obtiene las soluciones posibles para una partida.

- Busca palabras en el diccionario que se puedan formar con las letras disponibles.

Parameters

<i>available_letters</i>	Vector con las letras disponibles para jugar
<i>score_game</i>	Booleano: true para jugar por puntuación, false para jugar por longitud

Returns

Vector de strings con las palabras encontradas que cumplen los criterios

3.11.3.3 poderConstruir()

```
bool Solver::poderConstruir (
    string palabra,
    vector< char > available_letters )
```

Comprueba si una palabra se puede construir con un conjunto de letras.

Parameters

<i>palabra</i>	La palabra candidata
<i>available_letters</i>	Las letras disponibles en la mano

Returns

true si es posible construirla, false en caso contrario

3.11.3.4 puntosPalabra()

```
int Solver::puntosPalabra (
    string palabra )
```

Calcula la puntuación de una palabra.

Parameters

<i>palabra</i>	Palabra a evaluar
----------------	-------------------

Returns

Puntuación total basada en el [LetterSet](#) asociado

3.11.4 Member Data Documentation

3.11.4.1 dictionary

```
Dictionary Solver::dictionary [private]
```

3.11.4.2 ls

```
LetterSet Solver::ls [private]
```

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

- include/[solver.h](#)
- src/[solver.cpp](#)

Chapter 4

File Documentation

4.1 include/dictionary.h File Reference

```
#include <string>
#include <iostream>
#include <set>
#include <vector>
```

Include dependency graph for dictionary.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Dictionary](#)
- class [Dictionary::iterator](#)
Iterador del diccionario. Obtiene las palabras ordenadas alfabéticamente.
- class [Dictionary::const_iterator](#)
Iterador constante del diccionario. Obtiene las palabras ordenadas alfabéticamente.

4.2 dictionary.h

[Go to the documentation of this file.](#)

```
00001 //
00002 // Created by
00003 //
00004
00005 #ifndef DICTIONARY_H
00006 #define DICTIONARY_H
00007
00008
00009 #include <string>
00010 #include <iostream>
00011 #include <set>
00012
00013 #include <vector>
00014 using namespace std;
00015
00016 class Dictionary {
00017     private:
00018         set<string> words;
00019
00020     public:
00026         Dictionary();
00027
00031         ~Dictionary();
00032
```

```

00038     void clear();
00039
00045     unsigned int size() const;
00046
00051     bool empty() const;
00052
00059     bool exists(const string &val);
00060
00068     bool erase(const string &val);
00069
00079     friend istream &operator<>(istream &is, Dictionary &dic);
00080
00090     friend ostream &operator<>(ostream &os, const Dictionary &dic);
00091
00099     int getOccurrences(const char c) const;
00100
00105     int getTotalLetters() const;
00106
00110     class iterator{
00111     private:
00112         set<string>::iterator it;
00113     public:
00114         iterator ();
00115         iterator(set<string>::const_iterator);
00116         string operator *();
00117         iterator & operator ++();
00118         bool operator ==(const iterator &i);
00119         bool operator !=(const iterator &i);
00120
00121         friend class Dictionary;
00122     };
00123
00127     class const_iterator{
00128     private:
00129         set<string>::const_iterator it;
00130     public:
00131         const_iterator ();
00132         const_iterator(set<string>::const_iterator);
00133
00134         string operator *();
00135         const_iterator & operator ++();
00136         bool operator ==(const const_iterator &i);
00137         bool operator !=(const const_iterator &i);
00138
00139         friend class Dictionary;
00140     };
00141
00146     iterator find(const string & w);
00147
00153     vector<string> getWordsLength(int longitud);
00154
00162     pair<iterator,bool> insert(const string &val);
00163
00172     pair<iterator, iterator> range_prefix(const string &val);
00173
00177     iterator begin() ;
00178
00179     const_iterator begin() const;
00180
00184     iterator end() ;
00185
00186     const_iterator end() const;
00187
00188 };
00189 #endif //DICTIONARY_H

```

4.3 include/letters_bag.h File Reference

```

#include "letters_set.h"
#include <vector>
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include <string>

```

Include dependency graph for letters_bag.h: This graph shows which files directly or indirectly include this file:

Classes

- class [LettersBag](#)
TDA [LettersBag](#).
- class [LettersBag::iterator](#)
Iterador para recorrer la bolsa de letras.

4.4 letters_bag.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __LETTERS_BAG_H__
00002 #define __LETTERS_BAG_H__
00003
00004 #include "letters_set.h"
00005 #include <vector>
00006 #include <stdlib.h>
00007 #include <time.h>
00008 #include <algorithm> // para el find
00009 #include <string>
00010
00011 using namespace std;
00012
00013 class LettersBag {
00014     private:
00015         vector<char> bag;          // vector de letras
00016
00017     public:
00018         LettersBag(const LetterSet &);           // pasamos el numero de letras, y el conjunto
00019         // donde estan las letras
00020
00021         LettersBag(const string &);             // pasamos el numero de letras, y el conjunto
00022         // donde estan las letras
00023
00024         const char getLetter(int) const;         // conseguimos la letra
00025
00026         // void setLetter(const char);           // añadimos la letra a la bolsa (no le veo
00027         // mucho sentido que podamos añadir letras)
00028
00029         const string toString() const;          // mostramos por pantalla todas las letras de
00030         // la bolsa
00031
00032         unsigned size() const;
00033
00034         bool empty() const;
00035
00036         void clear();
00037
00038         void erase(const char);
00039
00040         class iterator{
00041             private:
00042                 vector<char>::iterator it;
00043
00044             public:
00045
00046                 iterator();
00047                 iterator(vector<char>::iterator);
00048
00049                 char & operator*(); // Devuelve una referencia al valor
00050                 iterator & operator++();
00051                 bool operator!=(const iterator &) const;
00052                 bool operator==(const iterator &) const;
00053
00054                 friend LettersBag;
00055         };
00056
00057         iterator begin();
00058
00059         iterator end();
00060     };
00061 #endif

```

4.5 include/letters_set.h File Reference

```
#include <iostream>
#include <string>
#include <fstream>
#include <map>
#include <stdlib.h>
```

Include dependency graph for letters_set.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [LetterInfo](#)
Estructura para almacenar información sobre una letra.
- class [LetterSet](#)
TDA [LetterSet](#).
- class [LetterSet::iterator](#)
Iterador para recorrer el conjunto de letras.
- class [LetterSet::const_iterator](#)
Iterador constante para recorrer el conjunto de letras.

4.6 letters_set.h

[Go to the documentation of this file.](#)

```
00001 #ifndef __LETTER_SET_H__
00002 #define __LETTER_SET_H__
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <fstream> // para la apertura del archivo y extraccion de letras
00007 #include <map>
00008 #include <stdlib.h> // para exit()
00009
00010 using namespace std;
00011
00015 struct LetterInfo{
00016     unsigned int repetitions;
00017     unsigned int score;
00018
00022     LetterInfo(): repetitions(0), score(0){}
00023
00030     LetterInfo(unsigned int reps, unsigned int score): repetitions(reps), score(score){};
00031 };
00032
00039 class LetterSet{
00040     private:
00041         map<char, LetterInfo> charSet; // mapeado de las letras
00042
00043     public:
00044         /*
00045          * @brief Constructor por defecto
00046          *
00047          * @param se pasara por referencia el nombre del archivo a abrir
00048          */
00049         LetterSet(const string &); // abriremos el archivo para la extraccion de las letras
00050
00057         const LetterInfo getLetterInfo(char) const;
00058
00059         // void setLetter(map<char,LetterInfo>); // inecesaria implementacion
00060
00065         bool empty() const;
00066
00071         unsigned size() const;
00072
00076         class iterator {
00077             private:
00078                 map<char, LetterInfo>::iterator it;
00079
```

```

00080         public:
00081
00082             iterator();
00083             iterator(map<char, LetterInfo>::iterator );
00084
00085             pair<const char, LetterInfo> & operator*(); // Devuelve una referencia al par
00086 clave-valor del mapa
00087             pair<const char, LetterInfo> * operator->();
00088
00089             iterator & operator++();
00089             bool operator!=(const iterator &) const;
00090             bool operator==(const iterator &) const;
00091
00092             friend class LetterSet;
00093     };
00094
00098     class const_iterator { // me salio un error y por eso lo implemente
00099     private:
00100         map<char, LetterInfo>::const_iterator it;
00101
00102     public:
00103
00104         const_iterator();
00105         const_iterator(map<char, LetterInfo>::const_iterator );
00106
00107         const pair<const char, LetterInfo> & operator*(); // Devuelve una referencia al par
00108 clave-valor del mapa
00109         const pair<const char, LetterInfo> * operator->();
00110
00111         const_iterator & operator++();
00112         bool operator!=(const const_iterator &) const;
00113         bool operator==(const const_iterator &) const;
00114
00115         friend class LetterSet;
00116     };
00120     iterator begin();
00121
00125     iterator end();
00126
00130     const_iterator begin() const;
00131
00135     const_iterator end() const;
00136 };
00137
00138 #endif

```

4.7 include/solver.h File Reference

```

#include <string>
#include <utility>
#include <vector>
#include "letters_set.h"
#include "dictionary.h"

```

Include dependency graph for solver.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Solver](#)
TDA [Solver](#).

4.8 solver.h

[Go to the documentation of this file.](#)

```

00001 #ifndef __SOLVER_H__
00002 #define __SOLVER_H__
00003
00004 #include <string>

```

```

00005 #include <utility>
00006 #include <vector>
00007
00008 #include "letters_set.h"
00009 #include "dictionary.h"
00010
00011 using namespace std;
00012
00019 class Solver{
00020     private:
00021         Dictionary dictionary;
00022         LetterSet ls;
00023
00024     public: // tienen que ser publicos para poder usarlos en el letras.cpp
00030         Solver(const Dictionary & dic, const LetterSet & letter) : dictionary(dic) , ls(letter){};
00031
00039         vector<string> getSolutions(const vector<char>& available_letters, bool score_game);
00040
00046         int puntosPalabra(string palabra);
00047
00054         bool poderConstruir(string palabra, vector<char> available_letters);
00055
00061         bool existe(string palabra);
00062 };
00063
00064 #endif // __SOLVER_H__

```

4.9 src/cantidad_letras.cpp File Reference

```

#include <fstream>
#include <iostream>
#include <cmath>
#include "dictionary.h"
#include "letters_set.h"

```

Include dependency graph for cantidad_letras.cpp:

4.10 src/cifras.cpp File Reference

```

#include <iostream>
#include <vector>
#include <set>
#include <cmath>
#include <climits>
#include <time.h>

```

Include dependency graph for cifras.cpp:

Classes

- struct [solucion](#)

Cifras Utilizamos dos funciones y un struct para calcular las operaciones necesarias para llegar a un número aleatorio de 3 cifras generado en el main, utilizando los números de un que se dan.

Functions

- vector< [solucion](#) > [GeneraOperaciones](#) ([solucion](#) actual, int n)
Calcula las posibles operaciones que se pueden conseguir, dadas una solución actual y un número que se le debe sumar, restar, multiplicar o dividir.
- [solucion](#) Cifras (multiset< int > S, int objetivo, [solucion](#) actual, [solucion](#) best)
Calcula la mejor solución para llegar al objetivo con los números del conjunto S.
- int [main](#) ()

4.10.1 Function Documentation

4.10.1.1 Cifras()

```
solucion Cifras (
    multiset< int > S,
    int objetivo,
    solucion actual,
    solucion best )
```

Calcula la mejor solución para llegar al objetivo con los números del conjunto S.

Parameters

<i>S</i>	es el conjunto de números con los que hay que operar para obtener la solución
<i>objetivo</i>	es el número que hay que alcanzar o acercarse al máximo
<i>actual</i>	es la solución actual, sobre la que seguiremos operando para llegar a la mejor
<i>best</i>	es la mejor solución hasta el momento

Returns

devuelve una solución con el valor exacto con el mínimo número de operaciones o la solución más cercana (también con el mínimo de operaciones)

4.10.1.2 GeneraOperaciones()

```
vector< solucion > GeneraOperaciones (
    solucion actual,
    int n )
```

Calcula las posibles operaciones que se pueden conseguir, dadas una solución actual y un número que se le debe sumar, restar, multiplicar o dividir.

Parameters

<i>actual</i>	es la solución parcial
<i>n</i>	es el número que operamos con actual

Returns

un vector de soluciones. Cada elemento será actual más/menos/por/entre n, según las operaciones que se puedan hacer (p.e. no puede haber números negativos)

4.10.1.3 main()

```
int main ( )
```

@Genera un multiset S de 6 elementos, escogidos al azar del conjunto C. Genera un número aleatorio de tres cifras al que hay que llegar. Pregunta al usuario su solución. Esta solución no tiene paréntesis, hace las operaciones de izquierda a derecha hasta que hay un valor inválido (al usuario se le recomienda usar ..). Después calcula con Cifras, la mejor solución y la muestra por pantalla

4.11 src/dictionary.cpp File Reference

```
#include "dictionary.h"
```

Include dependency graph for dictionary.cpp:

Functions

- `istream & operator>>` (`istream &is`, `Dictionary &dic`)
Sobrecarga del operador de entrada.
- `ostream & operator<<` (`ostream &os`, `const Dictionary &dic`)
Sobrecarga del operador de salida.

4.11.1 Function Documentation

4.11.1.1 `operator<<()`

```
ostream & operator<< (
    ostream & os,
    const Dictionary & dic )
```

Sobrecarga del operador de salida.

Permite imprimir el diccionario completo a un flujo de salida

Parameters

<i>os</i>	Flujo de salida, donde imprimir el diccionario
<i>dic</i>	Diccionario a imprimir

Returns

Flujo de salida, para poder encadenar el operador

4.11.1.2 `operator>>()`

```
istream & operator>> (
    istream & is,
    Dictionary & dic )
```

Sobrecarga del operador de entrada.

Permite leer las palabras de un fichero de texto e introducirlas en el diccionario

Parameters

<i>isz</i>	Flujo de entrada
<i>dic</i>	Diccionario a rellenar

Returns

Flujo de entrada para poder encadenar el operador

4.12 src/letras.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <time.h>
#include <set>
#include "dictionary.h"
#include "letters_bag.h"
#include "letters_set.h"
#include "solver.h"
Include dependency graph for letras.cpp:
```

Functions

- void [ModoPalabraMasLarga](#) ([Solver](#) &, const [LetterSet](#) &, int, string &, set< string > &, string &)
- void [ModoPalabraMayorPuntuacion](#) ([Solver](#) &, const [LetterSet](#) &, int, string &, set< string > &, string &)
- int [main](#) (int argc, char *argv[])

4.12.1 Function Documentation

4.12.1.1 main()

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

4.12.1.2 ModoPalabraMasLarga()

```
void ModoPalabraMasLarga (
    Solver & solver,
    const LetterSet & set_config,
    int CANTIDAD_LETRAS,
    string & solucion_user,
    set< string > & soluciones,
    string & mejor_solucion )
```

4.12.1.3 ModoPalabraMayorPuntuacion()

```
void ModoPalabraMayorPuntuacion (
    Solver & solver,
    const LetterSet & set_config,
    int CANTIDAD_LETRAS,
    string & solucion_user,
    set< string > & soluciones,
    string & mejor_solucion )
```

4.13 src/letters_bag.cpp File Reference

```
#include "letters_bag.h"
```

Include dependency graph for letters_bag.cpp:

4.14 src/letters_set.cpp File Reference

```
#include "letters_set.h"
```

Include dependency graph for letters_set.cpp:

4.15 src/solver.cpp File Reference

```
#include "solver.h"
#include <string>
#include <utility>
#include <vector>
#include "letters_set.h"
#include "dictionary.h"
```

Include dependency graph for solver.cpp:

4.16 src/testdiccionario.cpp File Reference

```
#include <fstream>
#include <iostream>
#include "../include/dictionary.h"
```

Include dependency graph for testdiccionario.cpp:

Functions

- int [main](#) (int argc, char *argv[])

4.16.1 Function Documentation

4.16.1.1 main()

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

Index

- ~Dictionary
 - Dictionary, [10](#)
- bag
 - LettersBag, [24](#)
- begin
 - Dictionary, [10](#)
 - LettersBag, [22](#)
 - LetterSet, [25](#)
- cantidad
 - solucion, [27](#)
- charSet
 - LetterSet, [27](#)
- Cifras
 - cifras.cpp, [39](#)
- cifras.cpp
 - Cifras, [39](#)
 - GeneraOperaciones, [39](#)
 - main, [39](#)
- clear
 - Dictionary, [10](#)
 - LettersBag, [22](#)
- const_iterator
 - Dictionary::const_iterator, [6](#)
 - LetterSet::const_iterator, [7](#)
- Dictionary, [9](#)
 - ~Dictionary, [10](#)
 - begin, [10](#)
 - clear, [10](#)
 - Dictionary, [10](#)
 - Dictionary::const_iterator, [6](#)
 - Dictionary::iterator, [16](#)
 - empty, [10](#)
 - end, [11](#)
 - erase, [11](#)
 - exists, [11](#)
 - find, [12](#)
 - getOccurrences, [12](#)
 - getTotalLetters, [12](#)
 - getWordsLength, [12](#)
 - insert, [12](#)
 - operator<<, [13](#)
 - operator>>, [14](#)
 - range_prefix, [13](#)
 - size, [13](#)
 - words, [14](#)
- dictionary
 - Solver, [31](#)
- dictionary.cpp
 - operator<<, [40](#)
 - operator>>, [40](#)
- Dictionary::const_iterator, [5](#)
 - const_iterator, [6](#)
 - Dictionary, [6](#)
 - it, [7](#)
 - operator!=, [6](#)
 - operator++, [6](#)
 - operator==, [6](#)
 - operator*, [6](#)
- Dictionary::iterator, [15](#)
 - Dictionary, [16](#)
 - it, [16](#)
 - iterator, [15](#)
 - operator!=, [16](#)
 - operator++, [16](#)
 - operator==, [16](#)
 - operator*, [16](#)
- empty
 - Dictionary, [10](#)
 - LettersBag, [23](#)
 - LetterSet, [26](#)
- end
 - Dictionary, [11](#)
 - LettersBag, [23](#)
 - LetterSet, [26](#)
- erase
 - Dictionary, [11](#)
 - LettersBag, [23](#)
- existe
 - Solver, [29](#)
- exists
 - Dictionary, [11](#)
- find
 - Dictionary, [12](#)
- GeneraOperaciones
 - cifras.cpp, [39](#)
- getLetter
 - LettersBag, [23](#)
- getLetterInfo
 - LetterSet, [26](#)
- getOccurrences
 - Dictionary, [12](#)
- getSolutions
 - Solver, [29](#)
- getTotalLetters

- Dictionary, 12
- getWordsLength
 - Dictionary, 12
- include/dictionary.h, 33
- include/letters_bag.h, 34, 35
- include/letters_set.h, 36
- include/solver.h, 37
- insert
 - Dictionary, 12
- it
 - Dictionary::const_iterator, 7
 - Dictionary::iterator, 16
 - LettersBag::iterator, 18
 - LetterSet::const_iterator, 8
 - LetterSet::iterator, 20
- iterator
 - Dictionary::iterator, 15
 - LettersBag::iterator, 17
 - LetterSet::iterator, 19
- letras.cpp
 - main, 41
 - ModoPalabraMasLarga, 41
 - ModoPalabraMayorPuntuacion, 41
- LetterInfo, 20
 - LetterInfo, 20
 - repetitions, 21
 - score, 21
- LettersBag, 21
 - bag, 24
 - begin, 22
 - clear, 22
 - empty, 23
 - end, 23
 - erase, 23
 - getLetter, 23
 - LettersBag, 22
 - LettersBag::iterator, 18
 - size, 24
 - toString, 24
- LettersBag::iterator, 16
 - it, 18
 - iterator, 17
 - LettersBag, 18
 - operator!=, 17
 - operator++, 17
 - operator==, 18
 - operator*, 17
- LetterSet, 24
 - begin, 25
 - charSet, 27
 - empty, 26
 - end, 26
 - getLetterInfo, 26
 - LetterSet, 25
 - LetterSet::const_iterator, 8
 - LetterSet::iterator, 20
 - size, 26
- LetterSet::const_iterator, 7
 - const_iterator, 7
 - it, 8
 - LetterSet, 8
 - operator!=, 8
 - operator++, 8
 - operator->, 8
 - operator==, 8
 - operator*, 8
- LetterSet::iterator, 18
 - it, 20
 - iterator, 19
 - LetterSet, 20
 - operator!=, 19
 - operator++, 19
 - operator->, 19
 - operator==, 19
 - operator*, 19
- Is
 - Solver, 31
- main
 - cifras.cpp, 39
 - letras.cpp, 41
 - testdiccionario.cpp, 42
- ModoPalabraMasLarga
 - letras.cpp, 41
- ModoPalabraMayorPuntuacion
 - letras.cpp, 41
- op
 - solucion, 27
- operator!=
 - Dictionary::const_iterator, 6
 - Dictionary::iterator, 16
 - LettersBag::iterator, 17
 - LetterSet::const_iterator, 8
 - LetterSet::iterator, 19
- operator<<
 - Dictionary, 13
 - dictionary.cpp, 40
- operator>>
 - Dictionary, 14
 - dictionary.cpp, 40
- operator++
 - Dictionary::const_iterator, 6
 - Dictionary::iterator, 16
 - LettersBag::iterator, 17
 - LetterSet::const_iterator, 8
 - LetterSet::iterator, 19
- operator->
 - LetterSet::const_iterator, 8
 - LetterSet::iterator, 19
- operator==
 - Dictionary::const_iterator, 6
 - Dictionary::iterator, 16
 - LettersBag::iterator, 18
 - LetterSet::const_iterator, 8
 - LetterSet::iterator, 19

- operator*
 - Dictionary::const_iterator, [6](#)
 - Dictionary::iterator, [16](#)
 - LettersBag::iterator, [17](#)
 - LetterSet::const_iterator, [8](#)
 - LetterSet::iterator, [19](#)
- poderConstruir
 - Solver, [29](#)
- puntosPalabra
 - Solver, [31](#)
- range_prefix
 - Dictionary, [13](#)
- repetitions
 - LetterInfo, [21](#)
- score
 - LetterInfo, [21](#)
- size
 - Dictionary, [13](#)
 - LettersBag, [24](#)
 - LetterSet, [26](#)
- solucion, [27](#)
 - cantidad, [27](#)
 - op, [27](#)
 - valor, [27](#)
- Solver, [28](#)
 - dictionary, [31](#)
 - existe, [29](#)
 - getSolutions, [29](#)
 - ls, [31](#)
 - poderConstruir, [29](#)
 - puntosPalabra, [31](#)
 - Solver, [28](#)
- src/cantidad_letras.cpp, [38](#)
- src/cifras.cpp, [38](#)
- src/dictionary.cpp, [40](#)
- src/letras.cpp, [41](#)
- src/letters_bag.cpp, [42](#)
- src/letters_set.cpp, [42](#)
- src/solver.cpp, [42](#)
- src/testdiccionario.cpp, [42](#)
- testdiccionario.cpp
 - main, [42](#)
- toString
 - LettersBag, [24](#)
- valor
 - solucion, [27](#)
- words
 - Dictionary, [14](#)