Kmer

5

Generated by Doxygen 1.9.6

Chapter 1

Deprecated List

Member ToLower (Kmer &kmer)

This function could go away in future versions

Member ToUpper (Kmer &kmer)

This function could go away in future versions

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 Kmer Class Reference

It represents a list of k consecutive nucleotides of a DNA or RNA sequence. Each nucleotide is represented with a character like 'A', 'C', 'G', 'T', 'U'.

`#include <Kmer.h>`

Public Member Functions

- Kmer (int k=5)

  It builds a Kmer object using a string with `k` characters (nucleotides). Each character will be set to the value `MISSING_NUCLEOTIDE`.
- Kmer (const std::string &text)

  It builds a Kmer object with the characters in the string `text` representing the list of nucleotides of the new Kmer.
- int getK () const

  Returns the number of nucleotides in this Kmer.
- int size () const

  Returns the number of nucleotides in this Kmer.
- std::string toString () const

  Returns a string with a list of characters, each one representing a nucleotide of this Kmer.
- const char & at (int index) const

  Gets a const reference to the character (nucleotide) at the given position.
- char & at (int index)

  Gets a reference to the character (nucleotide) at the given position.
- void toLower ()

  Converts uppercase letters in this Kmer to lowercase.
- void toUpper ()

  Converts lowercase letters in this Kmer to uppercase.
- void normalize (const std::string &validNucleotides)

  Normalizes this Kmer. That is, it converts all the characters to uppercase. Then, invalid characters are replaced by the MISSING_NUCLEOTIDE value.
- Kmer complementary (const std::string &nucleotides, const std::string &complementaryNucleotides) const

  Returns the complementary of this Kmer. For example, given the Kmer "TAGAC", the complementary is "ATCTG" (assuming that we use `nucleotides="ATGC"` and `complementaryNucleotides="TACG"`)
- void write (std::ostream &outputStream) const

Tells if the given Kmer argument can be found within this Kmer. For example the Kmer "tca" is within the Kmer "ttcaa" but is not within the Kmer "tctga".

- void read (std::istream &inputStream)

  Reads this object from the given input stream. It reads characters from the given input stream and put them in the text of this Kmer TODO: Revisar. Creo que debe escribirse entero + caracteres del string.

- const char & operator[ ] (int index) const

  Overloading of the [] operator for Kmer class. This method does not check if index is within the correct range.

- char & operator[ ] (int index)

  Overloading of the [] operator for Kmer classThis method does not check if index is within the correct range.

Static Public Attributes

- static const char MISSING_NUCLEOTIDE = '_'

Friends

- std::ostream & operator<< (std::ostream &os, const Kmer &kmer)

  Overloading of the stream insertion operator for Kmer class. It inserts the characters (nucleotides) of the given Kmer in the output string.

- std::istream & operator>> (std::istream &is, Kmer &kmer)

  Overloading of the stream extraction operator for Kmer class. It reads a list of characters from the input string that will set the list of nucleotides of the given Kmer.

4.1.1 Detailed Description

It represents a list of k consecutive nucleotides of a DNA or RNA sequence. Each nucleotide is represented with a character like 'A', 'C', 'G', 'T', 'U'.

Definition at line 27 of file Kmer.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Kmer() [1/2]

```
Kmer::Kmer (
            int k = 5 )
```

It builds a Kmer object using a string with k characters (nucleotides). Each character will be set to the value MISSING_NUCLEOTIDE.

Exceptions

std::invalid_argument	Throws an std::invalid_argument exception if k is less or equal than zero

Parameters

k	the number of nucleotides in this Kmer. It should be an integer greater than zero.

Definition at line 22 of file Kmer.cpp.

```
00022                 {
00023     if(k<1){
00024         throw std::invalid_argument(string("Kmer(int k): ") +
00025                 "invalid length " + to_string(k));
00026     }
00027     this->_text = string(k, MISSING_NUCLEOTIDE);
00028 }
```

4.1.2.2 Kmer() [2/2]

```
Kmer::Kmer (
            const std::string & text )
```

It builds a Kmer object with the characters in the string `text` representing the list of nucleotides of the new Kmer.

Exceptions

std::invalid_argument	Throws an std::invalid_argument exception if the given text is empty

Parameters

text	a string with the characters representing the nucleotides for the kmer. It should be a string with at least one character.

Definition at line 30 of file Kmer.cpp.

```
00030                                 {
00031     if(text.size()==0){
00032         throw std::invalid_argument(string("Kmer(const std::string& text): ") +
00033                 "text is an empty string ");
00034     }
00035     this->_text = string(text);
00036 }
```

4.1.3 Member Function Documentation

4.1.3.1 at() [1/2]

```
char & Kmer::at (
            int index )
```

Gets a reference to the character (nucleotide) at the given position.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws an std::out_of_range exception if the index is not in the range from 0 to k-1 (both included).

Returns

A reference to the character at the given position

Definition at line 60 of file Kmer.cpp.

```
00060                       {
00061     if(index<0 || index>=this->size()){
00062         throw std::out_of_range(string("char& Kmer::at(int index): ") +
00063                 "invalid position " + to_string(index));
00064     }
00065     else{
00066         return _text[index];
00067     }
00068 }
```

4.1.3.2 at() [2/2]

```
const char & Kmer::at (
            int index ) const
```

Gets a const reference to the character (nucleotide) at the given position.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws an std::out_of_range exception if the index is not in the range from 0 to k-1 (both included).

Returns

A const reference to the character at the given position

Definition at line 50 of file Kmer.cpp.

```
00050                                 {
00051     if(index<0 || index>=this->size()){
00052         throw std::out_of_range(string("const char& Kmer::at(int index) const: ") +
00053                 "invalid position " + to_string(index));
00054     }
00055     else{
00056         return _text[index];
00057     }
00058 }
```

4.1.3.3 complementary()

```
Kmer Kmer::complementary (
            const std::string & nucleotides,
            const std::string & complementaryNucleotides ) const
```

Returns the complementary of this Kmer. For example, given the Kmer "TAGAC", the complementary is "ATCTG" (assuming that we use `nucleotides="ATGC"` and `complementaryNucleotides="TACG"`)

Parameters

nucleotides	A string with the list of possible nucleotides
complementaryNucleotides	A string with the list of complementary nucleotides

Exceptions

std::invalid_argument	Throws an std::invalid_argument exception if the sizes of `nucleotides` and `complementaryNucleotides` are not the same

Returns

The complementary of this Kmer

Definition at line 106 of file Kmer.cpp.

```
00107                                                   {
00108
00109     if(nucleotides.size() != complementaryNucleotides.size()){
00110         throw std::invalid_argument(
00111             string("Kmer Kmer::complementary(const string& nucleotides, ") +
00112                 "const string& complementaryNucleotides) const:" +
00113                 " nucleotides and complementaryNucleotides have different lengths.");
00114     }
00115
00116     int pos;
00117     Kmer result(*this);
00118
00119     for(int i=0; i<result.size(); i++){
00120         pos =  nucleotides.find(result.at(i));
00121         if(pos !=string::npos ){ // if found
00122             result.at(i) = complementaryNucleotides.at(pos);
00123         }
00124     }
00125     return result;
00126 }
```

4.1.3.4 getK()

```
int Kmer::getK ( ) const
```

Returns the number of nucleotides in this Kmer.

Returns

the number of nucleotides in this Kmer

Definition at line 38 of file Kmer.cpp.

```
00038                       {
00039     return this->size();
00040 }
```

4.1.3.5 normalize()

```
void Kmer::normalize (
            const std::string & validNucleotides )
```

Normalizes this Kmer. That is, it converts all the characters to uppercase. Then, invalid characters are replaced by the MISSING_NUCLEOTIDE value.

Parameters

validNucleotides	a string with the list of characters (nucleotides) that should be considered as valid.

Definition at line 84 of file Kmer.cpp.

```
00084                                                {
00085 //    Version that does not use *this
00086 //    Kmer aux(this->toString());
00087 //    toLower(aux);
00088 //    for(int i=0; i<aux.size(); i++){
00089 //        if(!isValidNucleotide(aux.at(i), validNucleotides)){
00090 //            this->at(i) = Kmer.UNKNOWN_NUCLEOTIDE;
00091 //        }
00092 //        else{
00093 //            this->at(i) = aux.at(i);
00094 //        }
00095 //    }
00096
00097 //    ::ToUpper(*this);
00098     this->toUpper();
00099     for(int i=0; i<this->size(); i++){
00100         if(!IsValidNucleotide(this->at(i), validNucleotides)){
00101             this->at(i) = Kmer::MISSING_NUCLEOTIDE;
00102         }
00103     }
00104 }
```

4.1.3.6 operator[]() [1/2]

```
char & Kmer::operator[] (
            int index )
```

Overloading of the [] operator for Kmer classThis method does not check if index is within the correct range.

Parameters

index	index of the element (character)

Returns

A reference to the character at position `index`

Definition at line 144 of file Kmer.cpp.

```
00144                                {
00145     return _text[index];
00146 }
```

4.1.3.7 operator[]() [2/2]

```
const char & Kmer::operator[] (
            int index ) const
```

Overloading of the [] operator for Kmer class. This method does not check if index is within the correct range.

Parameters

index	index of the element (character)

Returns

A const reference to the character at position `index`

Definition at line 140 of file Kmer.cpp.

```
00140                                          {
00141     return _text[index];
00142 }
```

4.1.3.8 read()

```
void Kmer::read (
            std::istream & inputStream )
```

Reads this object from the given input stream. It reads characters from the given input stream and put them in the text of this Kmer TODO: Revisar. Creo que debe escribirse entero + caracteres del string.

Parameters

inputSstream	An input stream from which this object will be read

Definition at line 132 of file Kmer.cpp.

```
00132                                  {
00133     char* aux=new char[_text.size()+1];
00134
00135     inputStream.read(aux, _text.size()+1 );
00136     _text = aux;
00137     delete[] aux;
00138 }
```

4.1.3.9 size()

```
int Kmer::size ( ) const
```

Returns the number of nucleotides in this Kmer.

Returns

the number of nucleotides in this Kmer

Definition at line 42 of file Kmer.cpp.

```
00042                       {
00043     return this->_text.size();
00044 }
```

4.1.3.10 toLower()

```
void Kmer::toLower ( )
```

Converts uppercase letters in this Kmer to lowercase.

Definition at line 70 of file Kmer.cpp.

```
00070                       {
00071 //    ::ToLower(*this);
00072     for(int i=0; i<size(); i++){
00073        at(i) = tolower(at(i));
00074     }
00075 }
```

4.1.3.11 toString()

```
std::string Kmer::toString ( ) const
```

Returns a string with a list of characters, each one representing a nucleotide of this Kmer.

Returns

The text of this Kmer as a string object

Definition at line 46 of file Kmer.cpp.

```
00046                                 {
00047     return _text;
00048 }
```

4.1.3.12 toUpper()

```
void Kmer::toUpper ( )
```

Converts lowercase letters in this Kmer to uppercase.

Definition at line 77 of file Kmer.cpp.

```
00077                       {
00078 //    ::ToUpper(*this);
00079     for(int i=0; i<size(); i++){
00080        at(i) = toupper(at(i));
00081     }
00082 }
```

4.1.3.13 write()

```
void Kmer::write (
             std::ostream & outputStream ) const
```

Tells if the given Kmer argument can be found within this Kmer. For example the Kmer "tca" is within the Kmer "ttcaa" but is not within the Kmer "tctga".

Parameters

kmer	

Returns

Writes this object to the given output stream. All the characters in the string of this Kmer (including '\0') are sent to the ouput stream. TODO: Revisar. Creo que debe escribirse entero + caracteres del string

Parameters

outputStream	An output stream where this object will be written

Definition at line 128 of file Kmer.cpp.

```
00128                                           {
00129     outputStream.write(_text.c_str(),  _text.size()+1 );
00130 }
```

4.1.4 Friends And Related Function Documentation

4.1.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Kmer & kmer )  [friend]
```

Overloading of the stream insertion operator for Kmer class. It inserts the characters (nucleotides) of the given Kmer in the output string.

Parameters

os	The output stream to be used
kmer	the Kmer object

Returns

`os` A reference to the output stream

Definition at line 164 of file Kmer.cpp.

```
00164                                                   {
00165     os « kmer._text;
00166     return os;
00167 }
```

4.1.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            Kmer & kmer )  [friend]
```

Overloading of the stream extraction operator for Kmer class. It reads a list of characters from the input string that will set the list of nucleotides of the given Kmer.

Parameters

is	The input stream to be used
kmer	the Kmer object

Returns

`is` the input stream

Definition at line 169 of file Kmer.cpp.

```
00169                                                 {
00170     string chain;
00171     is » chain;
00172     kmer = Kmer(chain);
00173
00174     return is;
00175 }
```

4.1.5 Member Data Documentation

4.1.5.1 MISSING_NUCLEOTIDE

```
const char Kmer::MISSING_NUCLEOTIDE = '_'  [static]
```

A static const character representing an unknown nucleotide. It is used when we do not known which nucleotide we have in a given position of a Kmer

Definition at line 34 of file Kmer.h.

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

- include/Kmer.h
- src/Kmer.cpp

4.2 KmerCounter Class Reference

It is a helper class used to calculate the frequency of each kmer in a text file. It consists of a matrix of integers. Each element in the matrix contains the frequency of the kmer that is defined by its row and column: the kmer formed taking the nucleotides defined by the row and column of that element.

```
#include <KmerCounter.h>
```

Public Member Functions

- KmerCounter (int k=5, const std::string &validNucleotides=DEFAULT_VALID_NUCLEOTIDES)

  Constructor of the class. This object uses a matrix of integers that contains the frequency for each kmer. Each element of the matrix will be initialized with 0. The constructor also initializes the private data _k, _validNucleotides and _allNucleotides. _allNucleotides is initialized with the characters Kmer::MISSING_NUCLEOTIDE + validNucleotides.
- KmerCounter (const KmerCounter &orig)

  Copy constructor.
- ~KmerCounter ()

  Destructor.
- int getNumNucleotides () const

  Returns the number of nucleotides that can be part of a kmer, that is, the number of characters in the private data _allNucleotides. For example, if "_ACGT" are the set of all nucleotides, then this method will return 5.
- int getK () const

  Returns the number of nucleotides in each kmer.
- int getNumKmers () const

  Returns the number of different kmers that can be built using `_k` nucleotides (including the missing nucleotide)
- int getNumberActiveKmers () const

  Gets the number of kmers with a frequency greater than 0.
- void increaseFrequency (const Kmer &kmer, int frequency=1)

  Sets the frequency of the given kmer using the value provided with `frequency`.
- KmerCounter & operator= (const KmerCounter &orig)

  Overloading of the assignment operator.
- KmerCounter & operator+= (const KmerCounter &rkc)

  Overloading of the operator +=. It increases the current frequencies of the kmers of this object with the frequencies of the kmers of the given object.
- void calculateFrequencies (const char *fileName)

  Reads the given text file and calculates the frequencies of each kmer in that file. This method normalizes each found Kmer and then sum 1 at the corresponding element of the frequency matrix.
- Profile toProfile () const

  Builds a Profile object from this KmerCounter object. The Profile will contain the kmers and frequencies for those one with a frequency greater than 0. Note that this method does not provide build a Profile with an ordered vector of kmers. If you need an ordered vector of kmers, you must sort() the returned Profile after calling to this method.

Static Public Attributes

- static const char *const DEFAULT_VALID_NUCLEOTIDES ="ACGT"

4.2.1 Detailed Description

It is a helper class used to calculate the frequency of each kmer in a text file. It consists of a matrix of integers. Each element in the matrix contains the frequency of the kmer that is defined by its row and column: the kmer formed taking the nucleotides defined by the row and column of that element.

This class has a private data member string _validNucleotides to contain the set of possible nucleotides in a kmer and a private data member string _allNucleotides that contains the character that define the missing nucleotide (Kmer::MISSING_NUCLEOTIDE) plus the characters in _validNucleotides. Also, it contains a private data member int _k that defines the number of nucleotides in each kmer.

For example, if "ACGT" are the valid nucleotides, then _validNucleotides will be "ACGT", _allNucleotides will be "_ACGT". If the number of nucleotides in each kmer is 5, then _k will be 5. The first 3 nucleotides of each kmer will be associated to index the rows of the frequency matrix and the last 2 nucleotides to index the columns. For

example, for the kmer "ACATG", "ACA" will be used to index the row and "TG" to index the column. In this example, the first row corresponds to "___", the second to "__A", the third to "__C" and so on. The first column corresponds to "__", the second to "_A", the third to "_C" and so on.

When searching for kmers in a text file, if we find a character that does not belong to the set of valid nucleotides, it will be replaced by the character Kmer::MISSING_NUCLEOTIDE in the corresponding kmer

Definition at line 53 of file KmerCounter.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 KmerCounter() [1/2]

```
KmerCounter::KmerCounter (
            int k = 5,
            const std::string & validNucleotides = DEFAULT_VALID_NUCLEOTIDES )
```

Constructor of the class. This object uses a matrix of integers that contains the frequency for each kmer. Each element of the matrix will be initialized with 0. The constructor also initializes the private data _k, _validNucleotides and _allNucleotides. _allNucleotides is initialized with the characters Kmer::MISSING_NUCLEOTIDE + validNucleotides.

Parameters

k	The number of nucleotides in each kmer
validNucleotides	The set of nucleotides (characters) that are considered as part of a kmer.

4.2.2.2 KmerCounter() [2/2]

```
KmerCounter::KmerCounter (
            const KmerCounter & orig )
```

Copy constructor.

Parameters

orig	the KmerCounter object used as source for the copy

Definition at line 47 of file KmerCounter.cpp.

```
00047                                                {
00048     allocate(orig.getNumRows(), orig.getNumCols());
00049     copy(orig);
00050 }
```

4.2.2.3 ~KmerCounter()

```
KmerCounter::~KmerCounter ( )
```

Destructor.

Definition at line 52 of file KmerCounter.cpp.

```
00052                                {
00053     deallocate();
00054 }
```

4.2.3 Member Function Documentation

4.2.3.1 calculateFrequencies()

```
void KmerCounter::calculateFrequencies (
            const char * fileName )
```

Reads the given text file and calculates the frequencies of each kmer in that file. This method normalizes each found Kmer and then sum 1 at the corresponding element of the frequency matrix.

Parameters

fileName	The name of the file to process

Exceptions

std::ios_base::failure	Throws a std::ios_base::failure if the given file cannot be opened

Definition at line 139 of file KmerCounter.cpp.

```
00139                                                          {
00140     ifstream fe;
00141
00142     fe.open(fileName, ifstream::in);
00143     if (!fe) {
00144         throw std::ios_base::failure(
00145             string("void KmerCounter::calculateFrequencies(const char* fileName): ") +
00146                 "Error, opening file " + string(fileName));
00147     } else {
00148         int index = 0;
00149         int size;
00150         string inputString;
00151
00152         fe » inputString; // Read the DNA sequence
00153         size = inputString.size();
00154
00155         // Obtain the kmers
00156         while (size - index >= _k) { // exit if there are not enough characters to build a Kmer
00157             Kmer kmer = Kmer(inputString.substr(index, _k));
00158             kmer.normalize(this->_validNucleotides);
00159 //            cout « "KMER encontrado: " « kmer « endl;
00160             increaseFrequency(kmer);
00161             index++;
00162         }
00163         fe.close();
00164     }
00165 }
```

4.2.3.2 getK()

```
int KmerCounter::getK ( ) const
```

Returns the number of nucleotides in each kmer.

Returns

The number of nucleotides in each kmer

Definition at line 60 of file KmerCounter.cpp.

```
00060                           {
00061     return _k;
00062 }
```

4.2.3.3 getNumberActiveKmers()

```
int KmerCounter::getNumberActiveKmers ( ) const
```

Gets the number of kmers with a frequency greater than 0.

Returns

the number of kmers with a frequency greater than 0

Definition at line 68 of file KmerCounter.cpp.

```
00068                                           {
00069     int counter = 0;
00070     int nRows= getNumRows();
00071     int nCols = getNumCols();
00072
00073     for (int row = 0; row < nRows; ++row) {
00074         for (int column = 0; column < nCols; ++column) {
00075             if (_frequency[row][column] > 0)
00076                 counter++;
00077         }
00078     }
00079     return counter;
00080 }
```

4.2.3.4 getNumKmers()

```
int KmerCounter::getNumKmers ( ) const
```

Returns the number of different kmers that can be built using _k nucleotides (including the missing nucleotide)

Returns

The number of different kmers that can be built using _k nucleotides

Definition at line 64 of file KmerCounter.cpp.

```
00064                                   {
00065     return pow(getNumNucleotides(),_k);
00066 }
```

4.2.3.5 getNumNucleotides()

```
int KmerCounter::getNumNucleotides ( ) const
```

Returns the number of nucleotides that can be part of a kmer, that is, the number of characters in the private data _allNucleotides. For example, if "_ACGT" are the set of all nucleotides, then this method will return 5.

Returns

The number of nucleotides that can be part of a kmer

Definition at line 56 of file KmerCounter.cpp.

```
00056                                           {
00057     return _allNucleotides.size();
00058 }
```

4.2.3.6 increaseFrequency()

```
void KmerCounter::increaseFrequency (
            const Kmer & kmer,
            int frequency = 1 )
```

Sets the frequency of the given kmer using the value provided with `frequency`.

Parameters

kmer	The kmer in which the frequency will be set
frequency	The new frequency

Returns

true if the kmer was found in this object. false otherwise

Increases the current frequency of the given kmer using the value provided by `frequency`. If the argument `frequency` is not provided, then 1 is added to the current frequency of the kmer.

Exceptions

std::invalid_argument	This method throws an std::invalid_argument exception if the given kmer contains any invalid nucleotide

Parameters

kmer	The kmer in which the frequency will be modified
frequency	The quantity that will be added to the current frequency

Definition at line 94 of file KmerCounter.cpp.

```
00094                                                                   {
00095     int row, column;
```

```
00096
00097     this->getRowColumn(kmer, row, column);
00098     if(row<0 || column <0){
00099         throw std::invalid_argument(
00100                 string("void KmerCounter::increaseFrequency(const Kmer& kmer, int frequency): ") +
00101                 kmer.toString() + " contains invalid nucleotides");
00102     }
00103     else{
00104          _frequency[row][column] += frequency;
00105     }
00106 }
```

4.2.3.7 operator+=()

```
KmerCounter & KmerCounter::operator+= (
            const KmerCounter & rkc )
```

Overloading of the operator +=. It increases the current frequencies of the kmers of this object with the frequencies of the kmers of the given object.

Parameters

rkc	a KmerCounter object

Exceptions

std::invalid_argument	This method throws an std::invalid_argument exception if the given argument kmer has a different set of nucleotides or a different K (number of nucleotides in kmers).

Returns

A reference to this object

Definition at line 117 of file KmerCounter.cpp.

```
00117                                                       {
00118     if (_allNucleotides == rkc._allNucleotides && _k==rkc._k) {
00119         for (int f = 0; f < getNumRows(); f++)
00120             for (int c = 0; c < getNumCols(); c++)
00121                 _frequency[f][c] += rkc._frequency[f][c];
00122     }
00123     else{
00124         throw std::invalid_argument(
00125                 string("KmerCounter& KmerCounter::operator+=(const KmerCounter& rkc): ") +
00126                 " the given argument contains a different set nucleotides or a different K");
00127     }
00128     return *this;
00129 }
```

4.2.3.8 operator=()

```
KmerCounter & KmerCounter::operator= (
            const KmerCounter & orig )
```

Overloading of the assignment operator.

Parameters

orig	the KmerCounter object used as source for the assignment

Returns

A reference to this object

Definition at line 108 of file KmerCounter.cpp.

```
00108                                                          {
00109     if (this != &orig) {
00110         deallocate();
00111         allocate(orig.getNumRows(), orig.getNumCols());
00112         copy(orig);
00113     }
00114     return *this;
00115 }
```

4.2.3.9 toProfile()

```
Profile KmerCounter::toProfile ( ) const
```

Builds a Profile object from this KmerCounter object. The Profile will contain the kmers and frequencies for those one with a frequency greater than 0. Note that this method does not provide build a Profile with an ordered vector of kmers. If you need an ordered vector of kmers, you must sort() the returned Profile after calling to this method.

Returns

A Profile object from this KmerCounter object

Definition at line 167 of file KmerCounter.cpp.

```
00167                                       {
00168     int frequency;
00169     KmerFreq kmerFreq;
00170     Profile profile(this->getNumberActiveKmers());
00171 //    cout « "getNumberActiveKmers(): " « getNumberActiveKmers() « endl;
00172
00173     int nRows = this->getNumRows();
00174     int nCols = this->getNumCols();
00175     for (int pos = 0, row = 0; row < nRows; row++) {
00176         for (int col = 0; col < nCols; col++) {
00177             frequency = (*this)(row, col); // or this->_frequency[row][col];
00178             if (frequency > 0) {
00179                 Kmer kmer = this->getKmer(row, col);
00180
00181                 kmerFreq.setKmer(kmer);
00182                 kmerFreq.setFrequency(frequency);
00183 //                cout « "Kmer a insertar en Profile " « kmer « endl;
00184 //                cout « "profile size: " « profile.getSize() « endl;
00185                 profile.at(pos) = kmerFreq; // or profile[pos] = kmerFreq;
00186                 pos++;
00187             }
00188         }
00189     }
00190 //    profile.sort();
00191     return profile;
00192 }
```

4.2.4 Member Data Documentation

4.2.4.1 DEFAULT_VALID_NUCLEOTIDES

```
const char *const KmerCounter::DEFAULT_VALID_NUCLEOTIDES ="ACGT"  [static]
```

A const c-string with the set of characters that are considered as part of a word. Any other character will be considered a separator
Only lowercase characters are included in this string. This c-string is used in the constructor of this class, as the default value to assign to the field _validNucleotides

DEFAULT_VALID_NUCLEOTIDES is a c-string that contains the set of characters that will be considered as valid nucleotides.

The constructor of the class KmerCounter uses this c-string as a default parameter. It is possible to use a different c-string if that constructor is used with a different c-string

Definition at line 63 of file KmerCounter.h.

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

- include/KmerCounter.h
- src/KmerCounter.cpp

4.3 KmerFreq Class Reference

A pair formed by a Kmer object and a frequency (an int), that gives the frequency of a Kmer (times it appears) in a genoma.

```
#include <KmerFreq.h>
```

Public Member Functions

- KmerFreq ()

  Base constructor. It builds a KmerFreq object containing a Kmer with one nucleotide, the unknown nucleotide (Kmer::UNKNOWN_NUCLEOTIDE) and 0 as its frequency.
- const Kmer & getKmer () const

  Gets a const reference to the Kmer of this KmerFreq object.
- int getFrequency () const

  Gets the frequency of this KmerFreq object.
- void setKmer (const Kmer &kmer)

  Sets the Kmer of this KmerFreq object.
- void setFrequency (int frequency)

  Sets the frequency of this KmerFreq object.
- std::string toString () const

  Obtains a string with the string and frequency of the kmer in this object (separated by a whitespace).
- void write (std::ostream &outputStream) const

  Writes this object to the given output stream. It first writes the kmer of this object (using method Kmer::write(ostream&)) and them the bytes of the frequency (an int value) in binary format (using method ostream::write(const char s, streamsize n))*
- void read (std::istream &inputSstream)

  Reads this object from the given input stream. It first reads the Kmer of this object (using method Kmer::read(std::istream&) and then the bytes of the frequency (an int value) in binary format (using method istream::read(char s, streamsize n))*

Friends

- std::ostream & operator<< (std::ostream &os, const KmerFreq &kmerFreq)

  Overloading of the stream insertion operator for KmerFreq class.
- std::istream & operator>> (std::istream &is, KmerFreq &kmerFreq)

  Overloading of the stream extraction operator for KmerFreq class.

4.3.1 Detailed Description

A pair formed by a Kmer object and a frequency (an int), that gives the frequency of a Kmer (times it appears) in a genoma.

Definition at line 27 of file KmerFreq.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 KmerFreq()

```
KmerFreq::KmerFreq ( )
```

Base constructor. It builds a KmerFreq object containing a Kmer with one nucleotide, the unknown nucleotide (Kmer::UNKNOWN_NUCLEOTIDE) and 0 as its frequency.

Definition at line 23 of file KmerFreq.cpp.

```
00023                   :_kmer(string(1, Kmer::MISSING_NUCLEOTIDE)), _frequency(0) {
00024 }
```

4.3.3 Member Function Documentation

4.3.3.1 getFrequency()

```
int KmerFreq::getFrequency ( ) const
```

Gets the frequency of this KmerFreq object.

Returns

The frequency of this KmerFreq object

Definition at line 30 of file KmerFreq.cpp.

```
00030                                {
00031     return _frequency;
00032 }
```

4.3.3.2 getKmer()

```
const Kmer & KmerFreq::getKmer ( ) const
```

Gets a const reference to the Kmer of this KmerFreq object.

Returns

A const reference to the Kmer of this KmerFreq object

Definition at line 26 of file KmerFreq.cpp.

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

4.3.3.3 read()

```
void KmerFreq::read (
            std::istream & inputSstream )
```

Reads this object from the given input stream. It first reads the Kmer of this object (using method Kmer::read(std::istream&) and then the bytes of the frequency (an int value) in binary format (using method istream::read(char* s, streamsize n))

Parameters

inputSstream	An input stream from which this object will be read

Definition at line 93 of file KmerFreq.cpp.

```
00093                                          {
00094     _kmer.read(inputStream);
00095     inputStream.read(reinterpret_cast<char *> (&_frequency), sizeof(int));
00096 }
```

4.3.3.4 setFrequency()

```
void KmerFreq::setFrequency (
            int frequency )
```

Sets the frequency of this KmerFreq object.

Exceptions

std::out_of_range	if `frequency` is negative

Parameters

frequency	the new frequency value for this KmerFreq object

Definition at line 38 of file KmerFreq.cpp.

```
00038                                          {
00039     if(frequency<0){
00040         throw  std::out_of_range(string("void KmerFreq::setFrequency(int frequency): ") +
00041                 "invalid frequency " + to_string(frequency));
00042     }
00043     this->_frequency = frequency;
00044 }
```

4.3.3.5 setKmer()

```
void KmerFreq::setKmer (
            const Kmer & kmer )
```

Sets the Kmer of this KmerFreq object.

Parameters

kmer	The new Kmer value for this object

Definition at line 34 of file KmerFreq.cpp.

```
00034                                       {
00035     this->_kmer = kmer;
00036 }
```

4.3.3.6 toString()

```
string KmerFreq::toString ( ) const
```

Obtains a string with the string and frequency of the kmer in this object (separated by a whitespace).

Returns

A string with the nucleotide and frequency of the kmer in this object

Definition at line 46 of file KmerFreq.cpp.

```
00046                                  {
00047     return _kmer.toString() + " " + to_string(_frequency);
00048 }
```

4.3.3.7 write()

```
void KmerFreq::write (
            std::ostream & outputStream ) const
```

Writes this object to the given output stream. It first writes the kmer of this object (using method Kmer↩ ::write(ostream&)) and them the bytes of the frequency (an int value) in binary format (using method ostream↩ ::write(const char∗ s, streamsize n))

Parameters

outputStream	An output stream where this object will be written

Definition at line 88 of file KmerFreq.cpp.

```
00088                                                {
00089     _kmer.write(outputStream);
00090     outputStream.write(reinterpret_cast<const char *> (&_frequency), sizeof (int));
00091 }
```

4.3.4 Friends And Related Function Documentation

4.3.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const KmerFreq & kmerFreq )  [friend]
```

Overloading of the stream insertion operator for KmerFreq class.

Parameters

os	The output stream to be used
kmerFreq	the KmerFreq object

Returns

`os` A reference to the output stream

Definition at line 50 of file KmerFreq.cpp.

```
00050                                                                {
00051     os « kmerFreq.getKmer() « " " « kmerFreq.getFrequency();
00052     return os;
00053 }
```

4.3.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            KmerFreq & kmerFreq )  [friend]
```

Overloading of the stream extraction operator for KmerFreq class.

Parameters

is	The input stream to be used
kmerFreq	the KmerFreq object

Returns

`is` A reference to the input stream

Definition at line 55 of file KmerFreq.cpp.

```
00055                                                    {
00056     is » kmerFreq._kmer ;
00057     is » kmerFreq._frequency;
00058
00059     return is;
00060 }
```

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

- include/KmerFreq.h
- src/KmerFreq.cpp

4.4 Profile Class Reference

It defines a model (profile) for a given biological species. It contains a vector of pairs Kmer-frequency (objects of the class KmerFreq) and an identifier (string) of the profile.

`#include <Profile.h>`

Public Member Functions

- Profile ()

  Base constructor. It builds a Profile object with "unknown" as identifier, and an empty vector of pairs Kmer-frequency. The vector will have Kmer::INITIAL_CAPACITY as initial capacity.
- Profile (int size)

  It builds a Profile object with "unknown" as identifier, and a vector with a size of `size` pairs Kmer-frequency. The vector will also have `size` as initial capacity. Each pair will be initialized as Kmer::MISSING_NUCLEOTIDE for the Kmer and 0 for the frequency.
- Profile (const Profile &orig)

  Copy constructor.
- ~Profile ()

  Destructor.
- Profile & operator= (const Profile &orig)

  Overloading of the assignment operator for Profile class.
- const std::string & getProfileId () const

  Returns the identifier of this profile object.
- void setProfileId (const std::string &id)

  Sets a new identifier for this profile object.
- const KmerFreq & at (int index) const

  Gets a const reference to the KmerFreq at the given position of the vector in this object.
- KmerFreq & at (int index)

  Gets a reference to the KmerFreq at the given position of the vector in this object.
- int getSize () const

  Gets the number of KmerFreq objects in the vector of this object.
- int getCapacity () const

  Gets the capacity of the vector of KmerFreq objects.
- double getDistance (const Profile &otherProfile) const

Gets the distance between this Profile object (P_1) and the given argument object `otherProfile` (P_2). The distance between two Profiles P_1 and P_2 is calculated in the following way:

- int findKmer (const Kmer &kmer) const

  Searchs the given kmer in the list of kmers in this Profile. If found, it returns the position where it was found. If not, it returns -1. We consider that position 0 is the first kmer in the list of kmers and this->getSize()-1 the last kmer.

- std::string toString () const

  Obtains a string with the following content:

- void sort ()

  Sorts the vector of KmerFreq in decreasing order of frequency. If two KmerFreq objects have the same frequency, then the alphabetical order of the kmers of those objects will be considered (the object with a kmer that comes first alphabetically will appear first)

- void save (const char *fileName, char mode='t') const

  Saves this Profile object in the given file.

- void load (const char fileName[])

  Loads into this object the Profile object stored in the given file. Note that this method should remove any Kmer-frequency pairs that this object previously contained.

- void append (const KmerFreq &kmerFreq)

  Appends a copy of the given KmerFreq to this Profile object. If the kmer is found in this object, then its frequency is increased with the one of the given KmerFreq object. If not, a copy of the given KmerFreq object is appended to the end of the list of KmerFreq objects in this Profile.

- void normalize (const std::string &validNucleotides)

  Normalizes the Kmers of the vector of KmerFreq in this object. That is, for each Kmer in the vector, all its characters are converted to uppercase. Then, invalid characters are replaced by the MISSING_NUCLEOTIDE value.

- void deletePos (int pos)

  Deletes the KmerFreq object from the vector of KmerFreq in this object at the position `pos`. We consider that the first element has position 0, and the last element position size()-1.

- void zip (bool deleteMissing=false, int lowerBound=0)

  Deletes the KmerFreq objects from the vector of KmerFreq in this object which verifies one the following two criteria:

- KmerFreq & operator[ ] (int index) const

  Overloading of the [] operator for Profile class.

- KmerFreq & operator[ ] (int index)

  Overloading of the [] operator for Profile class.

- Profile & operator+= (const KmerFreq &kmerFreq)

  Overloading of the += operator with a KmerFreq parameter. It appends to this Profile object a copy of the given KmerFreq. If the kmer is found in this object, then its frequency is increased with the one of the given KmerFreq object. If not, a copy of the given KmerFreq object is appended to the end of the list of KmerFreq objects in this Profile.

- Profile & operator+= (const Profile &profile)

  Overloading of the += operator with a Profile parameter. For each kmer in the given Profile `profile`, if that kmer is found in this object, then its frequency is increased with the one in `profile`. If not, a copy of the kmer-pair is appended to the end of the list of KmerFreq objects in this Profile.

Friends

- std::ostream & operator<< (std::ostream &os, const Profile &profile)

  Overloading of the stream insertion operator for Profile class.

- std::istream & operator>> (std::istream &is, Profile &profile)

  Overloading of the stream extraction operator for Profile class. Note that this operator should remove any Kmer-frequency pairs that the argument Profile object previously contained.

4.4.1 Detailed Description

It defines a model (profile) for a given biological species. It contains a vector of pairs Kmer-frequency (objects of the class KmerFreq) and an identifier (string) of the profile.

Definition at line 28 of file Profile.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Profile() [1/3]

```
Profile::Profile ( )
```

Base constructor. It builds a Profile object with "unknown" as identifier, and an empty vector of pairs Kmer-frequency. The vector will have Kmer::INITIAL_CAPACITY as initial capacity.

Definition at line 27 of file Profile.cpp.

```
00027                    {
00028     _profileId="unknown";
00029     allocate(INITIAL_CAPACITY);
00030 }
```

4.4.2.2 Profile() [2/3]

```
Profile::Profile (
            int size )
```

It builds a Profile object with "unknown" as identifier, and a vector with a size of `size` pairs Kmer-frequency. The vector will also have `size` as initial capacity. Each pair will be initialized as Kmer::MISSING_NUCLEOTIDE for the Kmer and 0 for the frequency.

Exceptions

std::out_of_range	Throws a std::out_of_range exception if `size` < 0

Parameters

size	The size for the vector of kmers in this Profile

Definition at line 32 of file Profile.cpp.

```
00032                                {
00033     if(size < 0){
00034         throw std::out_of_range(string("Profile::Profile(int numberKmers): ") +
00035                 "invalid numberKmers=" + to_string(size));
00036     }
00037     _profileId="unknown";
00038     allocate(size);
00039     _size = size;
00040 }
```

4.4.2.3 Profile() [3/3]

```
Profile::Profile (
            const Profile & orig )
```

Copy constructor.

Parameters

orig	the Profile object used as source for the copy

Definition at line 42 of file Profile.cpp.

```
00042                                   {
00043     allocate(orig._capacity);
00044     copy(orig);
00045 }
```

4.4.2.4 ~Profile()

```
Profile::~Profile ( )
```

Destructor.

Definition at line 47 of file Profile.cpp.

```
00047                    {
00048     deallocate();
00049 }
```

4.4.3 Member Function Documentation

4.4.3.1 append()

```
void Profile::append (
            const KmerFreq & kmerFreq )
```

Appends a copy of the given KmerFreq to this Profile object. If the kmer is found in this object, then its frequency is increased with the one of the given KmerFreq object. If not, a copy of the given KmerFreq object is appended to the end of the list of KmerFreq objects in this Profile.

Parameters

kmerFreq	The KmerFreq to append to this object

Definition at line 242 of file Profile.cpp.

```
00242                                                {
00243     int pos = this->findKmer(kmerFreq.getKmer().toString());
00244     if (pos >= 0) { // If found
00245         this->at(pos).setFrequency(this->at(pos).getFrequency() +
00246                 kmerFreq.getFrequency());
00247     } else { // If not found
00248         if(this->_size == this->_capacity){ // If the vector is full
00249             this->reallocate(this->_capacity + BLOCK_SIZE);
00250         }
00251         this->_size++;
00252         this->at(this->_size-1) = kmerFreq;
00253     }
00254 }
```

4.4.3.2 at() [1/2]

```
KmerFreq & Profile::at (
            int index )
```

Gets a reference to the KmerFreq at the given position of the vector in this object.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws an std::out_of_range exception if the given index is not valid

Returns

A reference to the KmerFreq at the given position

Acceso seguro

Definition at line 76 of file Profile.cpp.

```
00076                                {
00077     if (0 <= index && index < getSize())
00078         return _vectorKmerFreq[index];
00079     else
00080         throw std::out_of_range(string("KmerFreq& Profile::at(int index): ") +
00081                 "invalid position " + to_string(index));
00082 }
```

4.4.3.3 at() [2/2]

```
const KmerFreq & Profile::at (
            int index ) const
```

Gets a const reference to the KmerFreq at the given position of the vector in this object.

Parameters

index	the position to consider

Exceptions

std::out_of_range	Throws an std::out_of_range exception if the given index is not valid

Returns

A const reference to the KmerFreq at the given position

Acceso seguro

Definition at line 68 of file Profile.cpp.

```
00068                                       {
00069     if (0 <= index && index < getSize())
00070         return _vectorKmerFreq[index];
00071     else
00072         throw std::out_of_range(string("const KmerFreq& Profile::at(int index) const: ") +
00073                 "invalid position " + to_string(index));
00074 }
```

4.4.3.4 deletePos()

```
void Profile::deletePos (
            int pos )
```

Deletes the KmerFreq object from the vector of KmerFreq in this object at the position `pos`. We consider that the first element has position 0, and the last element position size()-1.

Parameters

pos	The index of the position to be deleted.

Exceptions

std::out_of_range	Throws an std::out_of_range exception if `pos` is not in the range from 0 to size()-1 (both included).

Definition at line 265 of file Profile.cpp.

```
00265                                   {
00266     if(pos<0 || pos>=_size){
00267         throw std::out_of_range(
00268         string("Profile::deletePos(int pos): ") +
00269                 "invalid position " + to_string(pos));
00270     }
00271     for(int i=pos+1; i<_size; i++){
00272         _vectorKmerFreq[i-1] = _vectorKmerFreq[i];
00273     }
00274     _size--;
00275 }
```

4.4.3.5 findKmer()

```
int Profile::findKmer (
            const Kmer & kmer ) const
```

Searchs the given kmer in the list of kmers in this Profile. If found, it returns the position where it was found. If not, it returns -1. We consider that position 0 is the first kmer in the list of kmers and this->getSize()-1 the last kmer.

Parameters

kmer	A kmer

Returns

If found, it returns the position where the kmer was found. If not, it returns -1

Definition at line 116 of file Profile.cpp.

```
00116                                       { // const string& kmer) const  {
00117     for (int i = 0; i < getSize(); ++i) {
00118         if (kmer == this->at(i).getKmer()) { // or if (kmer == (*this)[i].getKmer()) {
00119             return i;
00120         }
00121     }
00122     return -1;
00123 }
```

4.4.3.6 getCapacity()

```
int Profile::getCapacity ( ) const
```

Gets the capacity of the vector of KmerFreq objects.

Returns

The capacity of the vector of KmerFreq objects

Definition at line 88 of file Profile.cpp.

```
00088                                {
00089     return _capacity;
00090 }
```

4.4.3.7 getDistance()

```
double Profile::getDistance (
            const Profile & otherProfile ) const
```

Gets the distance between this Profile object (P_1) and the given argument object `otherProfile` (P_2). The distance between two Profiles P_1 and P_2 is calculated in the following way:

$$d = \frac{\sum_{kmer_i(P_1)} |rank^{P_1}_{kmer_i(P_1)} - rank^{P_2}_{kmer_i(P_1)}|}{size(P_1) * size(P_2)},$$

where $kmer_i(p_j)$ is the kmer i of the Profile $p_j, j \in \{1, 2\}$ and $rank^{p_k}_{kmer_i(p_j)}$ is the ranking of the kmer i of the Profile $p_j, j \in \{1, 2\}$ in the Profile p_k.

The rank of a kmer is the position in which it appears in the list of KmerFreq. We consider 0 as the first position (rank equals to 0). When calculating $rank^{P_2}_{kmer_i(P_1)}$, if the kmer $kmer_i(P_1)$ does not appears in the Profile P_2 we consider that the rank is equals to the size of Profile P_2.

Parameters

otherProfile	A Profile object

Precondition

The list of kmers of this and otherProfile should be ordered in decreasing order of frequency. This is not checked in this method.

Exceptions

Throws	a std::invalid_argument exception if the implicit object (∗this) or the argument Profile object are empty, that is, they do not have any kmer.

Returns

The distance between this Profile object and the given argument `otherProfile`.

Definition at line 92 of file Profile.cpp.

```
00092                                                          {
00093     if(this->getSize() == 0 || otherProfile.getSize() == 0){ // CAMBIADO ESTE AÑO RESPECTO A 2022/2023
00094         throw std::invalid_argument(
00095            string("double Profile::getDistance(const Profile& otherProfile) const: ")
00096                + "*this or otherProfile do not have any kmer");
00097     }
00098     int posB;
00099     int posA = 0;
00100     double dist = 0.0;
00101
00102     for (int i = 0; i < getSize(); ++i) {
00103         posB = otherProfile.findKmer (this->at(i).getKmer()); // or posB =
      B.findKmer((*this)[i].getKmer());
00104         if (posB < 0) {
00105             posB = getSize();
00106 //            posB = otherProfile.getSize(); // CAMBIADO ESTE AÑO RESPECTO A 2022/2023
00107         }
00108
00109         dist += abs(posA - posB);
00110         posA++;
00111     }
00112     return dist / (getSize() * getSize());
00113 //    return dist / (getSize() *  otherProfile.getSize() );  // CAMBIADO ESTE AÑO RESPECTO A 2022/2023
00114 }
```

4.4.3.8 getProfileId()

```
const string & Profile::getProfileId ( ) const
```

Returns the identifier of this profile object.

Returns

A const reference to the identifier of this profile object

Definition at line 60 of file Profile.cpp.

```
00060                                       {
00061     return _profileId;
00062 }
```

4.4.3.9 getSize()

```
int Profile::getSize ( ) const
```

Gets the number of KmerFreq objects in the vector of this object.

Returns

The number of KmerFreq objects

Definition at line 84 of file Profile.cpp.

```
00084                               {
00085     return _size;
00086 }
```

4.4.3.10 load()

```
void Profile::load (
            const char fileName[] )
```

Loads into this object the Profile object stored in the given file. Note that this method should remove any Kmer-frequency pairs that this object previously contained.

Parameters

fileName	The name of the file where the Profile is stored

Exceptions

std::out_of_range	Throws a std::out_of_range exception if the number of kmers in the given file is negative.
std::ios_base::failure	Throws a std::ios_base::failure exception if the given file cannot be opened or if an error occurs while reading from the file
throw	std::invalid_argument Throws a std::invalid_argument exception if an invalid magic string is found in the given file

Definition at line 186 of file Profile.cpp.

```
00186                                       {
00187     ifstream inputStream;
00188     inputStream.open(fileName, ifstream::in | ios::binary);
00189     string magicString;
00190 
00191     if (inputStream) {
00192         inputStream » magicString;
00193         if (magicString == Profile::MAGIC_STRING_T) { // For text files
00194             inputStream » *this;
00195         } else if (magicString == Profile::MAGIC_STRING_B){ // Fo binary files
00196             int nKmers;
00197             string speciesId;
00198 
00199             inputStream » speciesId; // Read Species identifier
00200             this->setProfileId(speciesId);
00201 
00202             inputStream » nKmers; // Read number of kmers
00203             if (nKmers < 0) {
00204                 throw std::out_of_range(
00205                         string("void Profile::load(const char *fileName): ") +
```

```
00206                         "invalid number of kmers=" + to_string(nKmers));
00207             }
00208             inputStream.get(); // Read '\n' character that appears after nKmers
00209 
00210             deallocate();
00211 //            allocate(nKmers);
00212 
00213             KmerFreq kmerFreq;
00214             for(int i=0; i<nKmers; i++){
00215 //                _vectorKmerFreq[i].read(inputStream);
00216                 kmerFreq.read(inputStream);
00217                 this->append(kmerFreq);
00218             }
00219         }
00220         else {
00221             throw std::invalid_argument(
00222                 string("void Profile::load(const char *fileName): ") +
00223                 "the found magic string " + magicString + " in file " +
00224                 fileName + " is not valid ");
00225         }
00226         if (inputStream) {
00227             inputStream.close();
00228         }
00229         else{
00230             throw std::ios_base::failure(
00231                 string("void Profile::load(const char *fileName): ") +
00232                         "error reading from file " + fileName);
00233         }
00234     }
00235     else{
00236         throw std::ios_base::failure(
00237                 string("void Profile::load(const char *fileName): ") +
00238                         "error opening file " + fileName);
00239     }
00240 }
```

4.4.3.11 normalize()

```
void Profile::normalize (
            const std::string & validNucleotides )
```

Normalizes the Kmers of the vector of KmerFreq in this object. That is, for each Kmer in the vector, all its characters are converted to uppercase. Then, invalid characters are replaced by the MISSING_NUCLEOTIDE value.

Parameters

validNucleotides	a string with the list of characters (nucleotides) that should be considered as valid.

Definition at line 256 of file Profile.cpp.

```
00256                                                   {
00257     Kmer kmer;
00258     for(int i=0; i<_size; i++){
00259         kmer = _vectorKmerFreq[i].getKmer();
00260         kmer.normalize(validNucleotides);
00261         _vectorKmerFreq[i].setKmer(kmer);
00262     }
00263 }
```

4.4.3.12 operator+=() [1/2]

```
Profile & Profile::operator+= (
            const KmerFreq & kmerFreq )
```

Overloading of the += operator with a KmerFreq parameter. It appends to this Profile object a copy of the given KmerFreq. If the kmer is found in this object, then its frequency is increased with the one of the given KmerFreq object. If not, a copy of the given KmerFreq object is appended to the end of the list of KmerFreq objects in this Profile.

Parameters

kmerFreq	The KmerFreq object to append to this object

Returns

A reference to this object.

Definition at line 302 of file Profile.cpp.

```
00302                                                   {
00303     this->append(kmerFreq);
00304
00305     return *this;
00306 }
```

4.4.3.13 operator+=() [2/2]

```
Profile & Profile::operator+= (
            const Profile & profile )
```

Overloading of the += operator with a Profile parameter. For each kmer in the given Profile `profile`, if that kmer is found in this object, then its frequency is increased with the one in `profile`. If not, a copy of the kmer-pair is appended to the end of the list of KmerFreq objects in this Profile.

Parameters

profile	A Profile object

Returns

A reference to this object.

Definition at line 308 of file Profile.cpp.

```
00308                                                   {
00309     for (int i = 0; i < profile.getSize(); i++) {
00310         (*this)+=profile.at(i);
00311     }
00312
00313     return *this;
00314 }
```

4.4.3.14 operator=()

```
Profile & Profile::operator= (
            const Profile & orig )
```

Overloading of the assignment operator for Profile class.

Parameters

orig	the Profile object used as source for the assignment

Returns

A reference to this object

Definition at line 51 of file Profile.cpp.

```
00051                                                {
00052     if (this != &orig) {
00053         deallocate();
00054         allocate(orig._capacity );
00055         copy(orig);
00056     }
00057     return *this;
00058 }
```

4.4.3.15 operator[]() [1/2]

```
KmerFreq & Profile::operator[] (
            int index )
```

Overloading of the [] operator for Profile class.

Parameters

index	index of the element

Returns

A reference to the KmerFreq object at position `index`

Definition at line 298 of file Profile.cpp.

```
00298                                          {
00299     return _vectorKmerFreq[index];
00300 }
```

4.4.3.16 operator[]() [2/2]

```
KmerFreq & Profile::operator[] (
            int index ) const
```

Overloading of the [] operator for Profile class.

Parameters

index	index of the element

Returns

A reference to the KmerFreq object at position `index`

Definition at line 294 of file Profile.cpp.

```
00294                                                {
00295     return _vectorKmerFreq[index];
00296 }
```

4.4.3.17 save()

```
void Profile::save (
            const char * fileName,
            char mode = 't' ) const
```

Saves this Profile object in the given file.

Parameters

fileName	A c-string with the name of the file where this Profile object will be saved
mode	The mode to use to save this Profile object: 't' for text mode and 'b' for binary mode

Exceptions

std::ios_base::failure	Throws a std::ios_base::failure exception if the given file cannot be opened or if an error occurs while writing to the file

Definition at line 151 of file Profile.cpp.

```
00151                                                    {
00152     ofstream stream(fileName, ios::out | ios::binary);
00153 
00154     if (stream) {
00155         if (mode == 't') {
00156             stream << Profile::MAGIC_STRING_T << endl;
00157             stream << *this << endl;
00158         }
00159         else{ // mode == 'b'
00160             stream << Profile::MAGIC_STRING_B << endl;
00161             stream << this->getProfileId() << endl;
00162             stream << this->getSize() << endl;
00163 
00164 //            stream.write(reinterpret_cast<char *>(_vectorKmerFreq),
00165 //                    sizeof(KmerFreq)*this->getSize());
00166             for(int i=0; i<this->getSize(); i++){
00167                 _vectorKmerFreq[i].write(stream);
00168             }
00169         }
00170         if (stream)  {
00171             stream.close();
00172         }
00173         else{
00174             throw std::ios_base::failure(
00175                 string("void Profile::save(const char *fileName, char mode) const: ") +
00176                         "error writing to file " + fileName);
00177         }
00178     }
00179     else{
00180         throw std::ios_base::failure(
00181                 string("void Profile::save(const char *fileName, char mode) const: ") +
00182                         "error opening file " + fileName);
00183     }
00184 }
```

4.4.3.18 setProfileId()

```
void Profile::setProfileId (
            const std::string & id )
```

Sets a new identifier for this profile object.

Parameters

id	The new identifier

Definition at line 64 of file Profile.cpp.

```
00064                                                {
00065     _profileId = id;
00066 }
```

4.4.3.19 sort()

```
void Profile::sort ( )
```

Sorts the vector of KmerFreq in decreasing order of frequency. If two KmerFreq objects have the same frequency, then the alphabetical order of the kmers of those objects will be considered (the object with a kmer that comes first alphabetically will appear first)

Definition at line 135 of file Profile.cpp.

```
00135                       {
00136     KmerFreq aux;
00137     int pos;
00138     for (int i=0; i<getSize(); i++)  {
00139         pos = i;
00140         for (int j=i+1; j<getSize(); j++)
00141             if (this->at(j)>this->at(pos)) // or if ((*this)[j]>(*this)[pos])
00142                 pos = j;
00143         if (pos != i)  {
00144             aux = this->at(i); // or aux = (*this)[i];
00145             this->at(i) = this->at(pos);// or (*this)[i] = (*this)[pos];
00146             this->at(pos) = aux; // or (*this)[pos] = aux;
00147         }
00148     }
00149 }
```

4.4.3.20 toString()

```
std::string Profile::toString ( ) const
```

Obtains a string with the following content:

- In the first line, the profile identifier of this Profile
- In the second line, the number of kmers in this Profile
- In the following lines, each one of the pairs kmer-frequency (separated by a whitespace).

Returns

A string with the number of kmers and the list of pairs of kmer-frequency in the object

Definition at line 125 of file Profile.cpp.

```
00125                                   {
00126     string outputString = this->getProfileId() + "\n" +
00127             to_string(this->getSize()) + "\n";
00128
00129     for(int i=0; i<this->getSize(); i++){
00130         outputString += this->at(i).toString() + "\n";
00131     }
00132     return outputString;
00133 }
```

4.4.3.21 zip()

```
void Profile::zip (
            bool deleteMissing = false,
            int lowerBound = 0 )
```

Deletes the KmerFreq objects from the vector of KmerFreq in this object which verifies one the following two criteria:

1. The argument deleteMissing is true and the Kmer contains an unknown nucleotide
   (a) The frequency is less or equals to `lowerBound`.

Note that the number of elements in the argument array could be modified.

Parameters

deleteMissing	A bool value that indicates whether kmers with any unknown nucleotide should be removed. This parameter is false by default.
lowerBound	An integer value that defines which KmerFreq objects should be deleted from the vector of KmerFreq in this object. KmerFreq objects with a frequency less or equals to this value, are deleted. This parameter has zero as default value.

Definition at line 277 of file Profile.cpp.

```
00277                                                   {
00278     int pos;
00279
00280     pos = 0;
00281     while (pos < _size) {
00282         if ((deleteMissing &&
00283             _vectorKmerFreq[pos].getKmer().toString().find(
00284                 Kmer::MISSING_NUCLEOTIDE) != string::npos) ||
00285             _vectorKmerFreq[pos].getFrequency() <= lowerBound) {
00286
00287             deletePos(pos);
00288         } else {
00289             pos++;
00290         }
00291     }
00292 }
```

4.4.4 Friends And Related Function Documentation

4.4.4.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Profile & profile )  [friend]
```

Overloading of the stream insertion operator for Profile class.

Parameters

os	The output stream to be used
profile	the Profile object

Returns

`os` A reference to the output stream

4.4.4.2 operator>>

```
std::istream & operator>> (
            std::istream & is,
            Profile & profile )  [friend]
```

Overloading of the stream extraction operator for Profile class. Note that this operator should remove any Kmer-frequency pairs that the argument Profile object previously contained.

Exceptions

std::out_of_range	Throws a std::out_of_range if the number of kmers read from the file is negative.

Parameters

is	The input stream to be used
profile	the Profile object

Returns

`is` A reference to the input stream

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

- include/Profile.h
- src/Profile.cpp

Chapter 5

File Documentation

5.1 include/Kmer.h File Reference

```
#include <iostream>
#include <string>
```

Classes

- class Kmer

  It represents a list of k consecutive nucleotides of a DNA or RNA sequence. Each nucleotide is represented with a character like 'A', 'C', 'G', 'T', 'U'.

Functions

- bool IsValidNucleotide (char nucleotide, const std::string &validNucleotides)

  Checks if the given nucleotide is contained in `validNucleotides`. That is, if the given character can be considered as part of a genetic sequence.
- void ToLower (Kmer &kmer)

  Converts to lowercase the characters (nucleotides) of the given Kmer.
- void ToUpper (Kmer &kmer)

  Converts to uppercase the characters (nucleotides) of the given Kmer.
- std::ostream & operator<< (std::ostream &os, const Kmer &kmer)

  Overloading of the stream insertion operator for Kmer class. It inserts the characters (nucleotides) of the given Kmer in the output string.
- std::istream & operator>> (std::istream &is, Kmer &kmer)

  Overloading of the stream extraction operator for Kmer class. It reads a list of characters from the input string that will set the list of nucleotides of the given Kmer.
- bool operator> (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator > for Kmer class.
- bool operator< (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator < for Kmer class.
- bool operator== (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator == for Kmer class.
- bool operator!= (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator != for Kmer class.
- bool operator<= (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator <= for Kmer class.
- bool operator>= (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator >= for Kmer class.

5.1.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 23 October 2023, 12:25

Definition in file Kmer.h.

5.1.2 Function Documentation

5.1.2.1 IsValidNucleotide()

```
bool IsValidNucleotide (
            char nucleotide,
            const std::string & validNucleotides )
```

Checks if the given nucleotide is contained in `validNucleotides`. That is, if the given character can be considered as part of a genetic sequence.

Parameters

nucleotide	The nucleotide (a character) to check
validNucleotides	The set of characters that we consider as possible characters in a genetic sequence.

Returns

true if the given character is contained in `validNucleotides`; false otherwise

5.1.2.2 operator"!=()

```
bool operator!= (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator != for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if the two kmers contains different text; false otherwise

Definition at line 189 of file Kmer.cpp.

```
00189                                                 {
00190     return !(kmer1 == kmer2);
00191 }
```

5.1.2.3 operator<()

```
bool operator< (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator < for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 < kmer2; false otherwise

Definition at line 181 of file Kmer.cpp.

```
00181                                                 {
00182     return kmer2 > kmer1;
00183 }
```

5.1.2.4 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Kmer & kmer )
```

Overloading of the stream insertion operator for Kmer class. It inserts the characters (nucleotides) of the given Kmer in the output string.

Parameters

os	The output stream to be used
kmer	the Kmer object

Returns

os A reference to the output stream

Definition at line 174 of file Kmer.cpp.

```
00164                                                    {
00165     os « kmer._text;
00166     return os;
00167 }
```

5.1.2.5 operator<=()

```
bool operator<= (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator <= for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 <= kmer2; false otherwise

Definition at line 193 of file Kmer.cpp.

```
00193                                                    {
00194     return !(kmer1 > kmer2);
00195 }
```

5.1.2.6 operator==()

```
bool operator== (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator == for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if the two kmers contains the same text; false otherwise

Definition at line 185 of file Kmer.cpp.

```
00185                                                    {
00186     return !(kmer1<kmer2 || kmer1>kmer2);
00187 }
```

5.1.2.7 operator>()

```
bool operator> (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator > for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 > kmer2; false otherwise

Definition at line 177 of file Kmer.cpp.

```
00177                                                 {
00178     return kmer1.toString() > kmer2.toString();
00179 }
```

5.1.2.8 operator>=()

```
bool operator>= (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator >= for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 >= kmer2; false otherwise

Definition at line 197 of file Kmer.cpp.

```
00197                                                 {
00198     return !(kmer1 < kmer2);
00199 }
```

5.1.2.9 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            Kmer & kmer )
```

Overloading of the stream extraction operator for Kmer class. It reads a list of characters from the input string that will set the list of nucleotides of the given Kmer.

Parameters

is	The input stream to be used
kmer	the Kmer object

Returns

`is` the input stream

Definition at line 175 of file Kmer.cpp.

```
00169                                                 {
00170     string chain;
00171     is » chain;
00172     kmer = Kmer(chain);
00173
00174     return is;
00175 }
```

5.1.2.10 ToLower()

```
void ToLower (
            Kmer & kmer )
```

Converts to lowercase the characters (nucleotides) of the given Kmer.

Deprecated This function could go away in future versions

Parameters

kmer	A Kmer object

Definition at line 152 of file Kmer.cpp.

```
00152                           {
00153     for(int i=0; i<kmer.size(); i++){
00154        kmer.at(i) = tolower(kmer.at(i));
00155     }
00156 }
```

5.1.2.11 ToUpper()

```
void ToUpper (
            Kmer & kmer )
```

Converts to uppercase the characters (nucleotides) of the given Kmer.

Deprecated This function could go away in future versions

Parameters

kmer	A Kmer object

Definition at line 158 of file Kmer.cpp.

```
00158                                {
00159     for(int i=0; i<kmer.size(); i++){
00160        kmer.at(i) = toupper(kmer.at(i));
00161     }
00162 }
```

5.2 Kmer.h

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00015 #ifndef KMER_H
00016 #define KMER_H
00017
00018 #include <iostream>
00019 #include <string>
00020
00027 class Kmer {
00028 public:
00034     static const char MISSING_NUCLEOTIDE = '_';
00035
00046     Kmer(int k=5);
00047
00057     Kmer(const std::string& text);
00058
00063     int getK() const;
00064
00069     int size() const;
00070
00076     std::string toString() const;
00077
00086     const char& at(int index) const;
00087
00096     char& at(int index);
00097
00101     void toLower();
00102
00106     void toUpper();
00107
00116     void normalize(const std::string& validNucleotides);
00117
00130     Kmer complementary(const std::string& nucleotides,
00131          const std::string& complementaryNucleotides) const;
00132
00140 //    bool find(const Kmer& kmer) const;
00141
00148     void write(std::ostream& outputStream) const;
00149
00156     void read(std::istream& inputStream);
00157
00164     const char& operator[](int index) const;
00165
00172     char& operator[](int index);
00173
00174     friend std::ostream& operator<<(std::ostream& os, const Kmer& kmer);
00175     friend std::istream& operator>>(std::istream& is, Kmer& kmer);
00176
00177 private:
00182     std::string _text;
00183 }; // end class Kmer
00184
00195 bool IsValidNucleotide(char nucleotide, const std::string& validNucleotides);
00196
00202 void ToLower(Kmer& kmer);
00203
00209 void ToUpper(Kmer& kmer);
00210
00211
00219 std::ostream& operator<<(std::ostream& os, const Kmer& kmer);
```

```
00220
00229 std::istream& operator»(std::istream& is, Kmer& kmer);
00230
00237 bool operator>(const Kmer& kmer1, const Kmer& kmer2);
00238
00245 bool operator<(const Kmer& kmer1, const Kmer& kmer2);
00246
00253 bool operator==(const Kmer& kmer1, const Kmer& kmer2);
00254
00255
00262 bool operator!=(const Kmer& kmer1, const Kmer& kmer2);
00263
00270 bool operator<=(const Kmer& kmer1, const Kmer& kmer2);
00271
00278 bool operator>=(const Kmer& kmer1, const Kmer& kmer2);
00279
00280 #endif /* KMER_H */
00281
```

5.3 KmerCounter.h

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00006 /*
00007  * @file:   KmerCounter.h
00008  * @author Silvia Acid Carrillo <acid@decsai.ugr.es>
00009  * @author Andrés Cano Utrera <acu@decsai.ugr.es>
00010  * @author Luis Castillo Vidal <L.Castillo@decsai.ugr.es>
00011  *
00012  * Created on 7 November 2023, 14:00
00013  */
00014
00015 #ifndef KMER_COUNTER_H
00016 #define KMER_COUNTER_H
00017
00018 #include <string>
00019
00020 #include "Profile.h"
00021
00053 class KmerCounter {
00054 public:
00055
00063     static const char* const DEFAULT_VALID_NUCLEOTIDES;
00064
00076     KmerCounter(int k=5,
00077        const std::string& validNucleotides = DEFAULT_VALID_NUCLEOTIDES);
00078
00083     KmerCounter(const KmerCounter & orig);
00084
00088     ~KmerCounter();
00089
00097     int getNumNucleotides() const;
00098
00103     int getK() const;
00104
00111     int getNumKmers() const;
00112
00113
00118     int getNumberActiveKmers() const;
00119
00127 //    bool setFrequency(const Kmer& kmer, int frequency);
00128
00129
00140     void increaseFrequency(const Kmer& kmer, int frequency = 1);
00141
00147     KmerCounter& operator=(const KmerCounter & orig);
00148
00149
00161     KmerCounter& operator+=(const KmerCounter & rkc);
00162
00171     void calculateFrequencies(const char* fileName);
00172
00182     Profile toProfile() const;
00183
00191 //    void fromProfile(const Profile & profile);
00192 private:
00193     int** _frequency;
00194
00195     int _k;
00196
```

```
00202     std::string _validNucleotides;
00203
00208     std::string _allNucleotides;
00209
00214     int getNumRows() const;
00215
00220     int getNumCols() const;
00221
00232     int getIndex(const std::string& kmer) const;
00233
00245     std::string getInvertedIndex(int index, int nCharacters) const;
00246
00256     void getRowColumn(const Kmer &kmer, int& row, int& column) const;
00257
00267     Kmer getKmer(int row, int column) const;
00268
00273     void initFrequencies();
00274
00284     void allocate(int nRows, int nCols);
00285
00289     void deallocate();
00290
00291
00302     void copy(const KmerCounter & other);
00303
00311     const int& operator()(int row, int column) const;
00312
00320     int& operator()(int row, int column);
00321 };
00322
00323 #endif /* KMER_COUNTER_H */
```

5.4 KmerFreq.h

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00006 /*
00007  * @file   KmerFreq.h
00008  * @author Silvia Acid Carrillo <acid@decsai.ugr.es>
00009  * @author Andrés Cano Utrera <acu@decsai.ugr.es>
00010  * @author Luis Castillo Vidal <L.Castillo@decsai.ugr.es>
00011  * @author Javier Martínez Baena jbaena@ugr.es
00012  *
00013  * Created on 25 October 2023, 19:53
00014  *
00015  */
00016
00017 #ifndef KMER_FREQ_H
00018 #define KMER_FREQ_H
00019
00020 #include "Kmer.h"
00021
00027 class KmerFreq {
00028 public:
00034     KmerFreq();
00035
00040     const Kmer& getKmer() const;
00041
00046     int getFrequency() const;
00047
00052     void setKmer(const Kmer& kmer);
00053
00059     void setFrequency(int frequency);
00060
00067     std::string toString() const;
00068
00076     void write(std::ostream& outputStream) const;
00077
00085     void read(std::istream& inputSstream);
00086
00087     friend std::ostream& operator«(std::ostream& os, const KmerFreq& kmerFreq);
00088     friend std::istream& operator»(std::istream& is, KmerFreq& kmerFreq);
00089
00090 private:
00091     Kmer _kmer;
00092     int _frequency;
00093 }; // end class KmerFreq
00094
00101 std::ostream& operator«(std::ostream& os, const KmerFreq& kmerFreq);
00102
```

```
00109 std::istream& operator»(std::istream& is, KmerFreq& kmerFreq);
00110
00119 bool operator>(const KmerFreq &kmerFreq1, const KmerFreq &kmerFreq2);
00120
00127 bool operator<(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2);
00128
00136 bool operator==(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2);
00137
00145 bool operator!=(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2);
00146
00153 bool operator<=(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2);
00154
00161 bool operator>=(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2);
00162
00163 #endif /* KMER_FREQ_H */
00164
```

5.5 include/Profile.h File Reference

```
#include <iostream>
#include "KmerFreq.h"
```

Classes

- class Profile

  It defines a model (profile) for a given biological species. It contains a vector of pairs Kmer-frequency (objects of the class KmerFreq) and an identifier (string) of the profile.

Functions

- std::ostream & operator<< (std::ostream &os, const Profile &profile)

  Overloading of the stream insertion operator for Profile class.
- std::istream & operator>> (std::istream &is, Profile &profile)

  Overloading of the stream extraction operator for Profile class. Note that this operator should remove any Kmer-frequency pairs that the argument Profile object previously contained.

5.5.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 31 October 2023, 9:40

Definition in file Profile.h.

5.5.2 Function Documentation

5.5.2.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Profile & profile )
```

Overloading of the stream insertion operator for Profile class.

Parameters

os	The output stream to be used
profile	the Profile object

Returns

`os` A reference to the output stream

5.5.2.2 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            Profile & profile )
```

Overloading of the stream extraction operator for Profile class. Note that this operator should remove any Kmer-frequency pairs that the argument Profile object previously contained.

Exceptions

std::out_of_range	Throws a std::out_of_range if the number of kmers read from the file is negative.

Parameters

is	The input stream to be used
profile	the Profile object

Returns

`is` A reference to the input stream

5.6 Profile.h

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00015 #ifndef PROFILE_H
00016 #define PROFILE_H
00017
00018
00019 #include <iostream>
00020 #include "KmerFreq.h"
00021
00028 class Profile {
00029 public:
00030
00036     Profile();
00037
00048     Profile(int size);
00049
```

```
00054     Profile(const Profile& orig);
00055
00059     ~Profile();
00060
00066     Profile& operator=(const Profile& orig);
00067
00072     const std::string& getProfileId() const;
00073
00078     void setProfileId(const std::string& id);
00079
00080
00089     const KmerFreq& at(int index) const;
00090
00099     KmerFreq& at(int index);
00100
00105     int getSize() const;
00106
00111     int getCapacity() const;
00112
00142     double getDistance(const Profile& otherProfile) const;
00143
00153     int findKmer(const Kmer& kmer) const;
00154
00164     std::string toString() const;
00165
00172     void sort();
00173
00184     void save(const char *fileName, char mode = 't') const;
00185
00199     void load(const char fileName[]);
00200
00209     void append(const KmerFreq& kmerFreq);
00210
00219     void normalize(const std::string& validNucleotides);
00220
00229     void deletePos(int pos);
00230
00247     void zip(bool deleteMissing=false, int lowerBound = 0);
00248
00254     KmerFreq& operator[](int index) const;
00255
00261     KmerFreq& operator[](int index);
00262
00273     Profile& operator+=(const KmerFreq& kmerFreq);
00274
00275
00285     Profile& operator+=(const Profile& profile);
00286
00287     friend std::ostream & operator«(std::ostream & os, const Profile & profile);
00288     friend std::istream & operator»(std::istream & is, Profile & profile);
00289
00290 private:
00291     std::string _profileId;
00292     KmerFreq* _vectorKmerFreq;
00293     int _size;
00294     int _capacity;
00295
00296     static const int INITIAL_CAPACITY=10;
00297     static const int BLOCK_SIZE=20;
00298
00299     static const std::string MAGIC_STRING_T;
00300     static const std::string MAGIC_STRING_B;
00301
00306 //    void setSize(int size);
00307
00314     void allocate(int capacity);
00315
00331     void reallocate(int newCapacity);
00332
00338     void deallocate();
00339
00349     void copy(const Profile& otherProfile);
00350 };
00351
00358 std::ostream & operator«(std::ostream & os, const Profile & profile);
00359
00370 std::istream & operator»(std::istream & is, Profile & profile);
00371
00372
00373 #endif /* PROFILE_H */
```

5.7 src/CLASSIFY.cpp File Reference

```
#include <iostream>
#include "KmerCounter.h"
#include "Profile.h"
```

Functions

- void showSpanishHelp (ostream &outputStream)
- void showEnglishHelp (ostream &outputStream)
- int main (int argc, char ∗argv[ ])

5.7.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 10 November 2023, 13:00

Definition in file CLASSIFY.cpp.

5.7.2 Function Documentation

5.7.2.1 main()

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

This program print the profile identifier of the closest profile for an input text file (<text.txt>) among the set of provided models: <lang1.bgr>, <lang2.bgr>, ...

Running example:

CLASSIFY <text.txt> <lang1.bgr> [<lang2.bgr> <lang3.bgr> ...]

Parameters

argc	The number of command line parameters
argv	The vector of command line parameters (cstrings)

Returns

0 If there is no error; a value $>$ 0 if error

Definition at line 51 of file CLASSIFY.cpp.

```
00051                                   {
00052     if (argc < 3) {
00053         showEnglishHelp(cerr);
00054         return 1;
00055     } else {
00056         KmerCounter kmerCounter;
00057         kmerCounter.calculateFrequencies(argv[1]);
00058         Profile profile;
00059         Profile unknownProfile=kmerCounter.toProfile();
00060 
00061         profile.load(argv[2]);
00062         string profileId = profile.getProfileId();
00063         double minDistance = unknownProfile.getDistance(profile);
00064         double currentDistance;
00065         for (int i = 3; i < argc; i++) {
00066             profile.load(argv[i]);
00067             currentDistance = unknownProfile.getDistance(profile);
00068             if (currentDistance < minDistance) {
00069                 minDistance = currentDistance;
00070                 profileId = profile.getProfileId();
00071             }
00072         }
00073         cout « "Final decision: profile " « profileId « " with a distance of " « minDistance « endl;
00074     }
00075     return 0;
00076 }
```

5.7.2.2 showEnglishHelp()

```
void showEnglishHelp (
             ostream & outputStream )
```

Shows help about the use of this program in the given output stream

Parameters

outputStream	The output stream where the help will be shown (for example, cout, cerr, etc)

Definition at line 33 of file CLASSIFY.cpp.

```
00033                                           {
00034     outputStream « "Error, run with the following parameters:" « endl;
00035     outputStream « "CLASSIFY <text.txt> <lang1.bgr> [<lang2.bgr> <lang3.bgr> ....]" « endl;
00036     outputStream « "          Obtains the identifier of the closest profile to the input text file" «
      endl;
00037     outputStream « endl;
00038 }
```

5.7.2.3 showSpanishHelp()

```
void showSpanishHelp (
             ostream & outputStream )
```

Definition at line 21 of file CLASSIFY.cpp.

```
00021                                           {
00022     outputStream « "Error, ejecute con los siguientes parámetros:" « endl;
00023     outputStream « "CLASSIFY  <texto.txt> <leng1.bgr> [<leng2.bgr> <leng3.bgr> ....]" « endl;
00024     outputStream « "          Devuelve el lenguaje más cercano al texto de entrada" « endl;
00025     outputStream « endl;
00026 }
```

5.8 CLASSIFY.cpp

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Profile5
00003  * Curso 2023/2024
00004  */
00005
00015 #include <iostream>
00016 #include "KmerCounter.h"
00017 #include "Profile.h"
00018
00019 using namespace std;
00020
00021 void showSpanishHelp(ostream& outputStream) {
00022     outputStream « "Error, ejecute con los siguientes parámetros:" « endl;
00023     outputStream « "CLASSIFY  <texto.txt> <leng1.bgr> [<leng2.bgr> <leng3.bgr> ....]" « endl;
00024     outputStream « "          Devuelve el lenguaje más cercano al texto de entrada" « endl;
00025     outputStream « endl;
00026 }
00027
00033 void showEnglishHelp(ostream& outputStream) {
00034     outputStream « "Error, run with the following parameters:" « endl;
00035     outputStream « "CLASSIFY <text.txt> <lang1.bgr> [<lang2.bgr> <lang3.bgr> ....]" « endl;
00036     outputStream « "          Obtains the identifier of the closest profile to the input text file" «
      endl;
00037     outputStream « endl;
00038 }
00039
00051 int main(int argc, char *argv[]) {
00052     if (argc < 3) {
00053         showEnglishHelp(cerr);
00054         return 1;
00055     } else {
00056         KmerCounter kmerCounter;
00057         kmerCounter.calculateFrequencies(argv[1]);
00058         Profile profile;
00059         Profile unknownProfile=kmerCounter.toProfile();
00060
00061         profile.load(argv[2]);
00062         string profileId = profile.getProfileId();
00063         double minDistance = unknownProfile.getDistance(profile);
00064         double currentDistance;
00065         for (int i = 3; i < argc; i++) {
00066             profile.load(argv[i]);
00067             currentDistance = unknownProfile.getDistance(profile);
00068             if (currentDistance < minDistance) {
00069                 minDistance = currentDistance;
00070                 profileId = profile.getProfileId();
00071             }
00072         }
00073         cout « "Final decision: profile " « profileId « " with a distance of " « minDistance « endl;
00074     }
00075     return 0;
00076 }
00077
```

5.9 src/JOIN.cpp File Reference

```
#include <iostream>
#include <cstring>
#include "BigramCounter.h"
#include "Language.h"
```

Functions

- void showEnglishHelp (ostream &outputStream)
- int main (int argc, char *argv[])

5.9.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 29 January 2023, 11:00

Definition in file JOIN.cpp.

5.9.2 Function Documentation

5.9.2.1 main()

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

This program reads an undefined number of Language objects from the files passed as parameters to main(). It obtains as result the union of all the input Language objects. The result is then sorted by decreasing order of frequency and alphabetical order of bigrams when there is any tie in frequencies. Finally, the resulting Language is saved in an output file. The program must have at least an input file. Running example:

JOIN [-t|-b] [-o <outputFile.bgr>] <file1.bgr> [<file2.bgr> ... <filen.bgr>]

Parameters

argc	The number of command line parameters
argv	The vector of command line parameters (cstrings)

Returns

0 If there is no error; a value > 0 if error

Definition at line 52 of file JOIN.cpp.

```
00052                                   {
00053     Language inputLanguage, inputLanguageAux;
00054     string outputFileName = "output.bgr";
00055     int nInputFiles;
00056     int firstInputFileArg = -1, // position in argv of the first input file name
00057         lastInputFileArg = -1; // position in argv of the last input file name
00058
00059     char mode = 't'; // text mode ('t') or binary ('b')
00060     bool hasBeenReadInitialParameters = false; // true if all the parameters with - has been read
00061
00062     for (int i = 1; i < argc && !hasBeenReadInitialParameters; ++i) { // Loop to process the main()
    parameters
00063         if (argv[i][0] == '-' && !hasBeenReadInitialParameters) {
00064             if(strcmp(argv[i], "-t") == 0){
00065                 mode = 't';
```

```
00066             }
00067             else if(strcmp(argv[i], "-b") == 0){
00068                 mode = 'b';
00069             }
00070             else if (strcmp(argv[i], "-o") == 0) {
00071                 if ((i + 1) < argc) { // If at least another parameter is provided (output file name)
00072                     outputFileName = argv[i + 1];
00073                     ++i;
00074                 }
00075             }
00076             else{
00077                 showEnglishHelp(cerr);
00078                 return 1;
00079             }
00080         }
00081         else {
00082             hasBeenReadInitialParameters = true;
00083             firstInputFileArg = i;
00084         }
00085     }
00086     lastInputFileArg = argc-1;
00087     if (firstInputFileArg < 0) {
00088         showEnglishHelp(cerr);
00089         return 1;
00090     }
00091
00092     inputLanguage.load(argv[firstInputFileArg]);
00093     for (int i = firstInputFileArg+1; i <= lastInputFileArg; i++) {
00094         inputLanguageAux.load(argv[i]);
00095         if(inputLanguageAux.getLanguageId() == inputLanguage.getLanguageId()){
00096             inputLanguage+=inputLanguageAux;
00097         }
00098     }
00099     inputLanguage.sort();
00100     inputLanguage.save(outputFileName.c_str(),mode);
00101     return 0;
00102 }
```

5.9.2.2 showEnglishHelp()

```
void showEnglishHelp (
             ostream & outputStream )
```

Shows help about the use of this program in the given output stream

Parameters

outputStream	The output stream where the help will be shown (for example, cout, cerr, etc)

Definition at line 28 of file JOIN.cpp.

```
00028                                          {
00029     outputStream « "Error, run with the following parameters:" « endl;
00030     outputStream « "JOIN [-t|-b] [-o <outputFile.bgr>] <file1.bgr> [<file2.bgr> ... <filen.bgr>] " «
      endl;
00031     outputStream « "       join the Language files <file1.bgr> <file2.bgr> ... into <outputFile.bgr>"
      « endl;
00032     outputStream « endl;
00033     outputStream « "Parameters:" « endl;
00034     outputStream « "-t|-b: text mode or binary mode for the output file (-t by default)" « endl;
00035     outputStream « "-o <outputFile.bgr>: name of the output file (output.bgr by default)" « endl;
00036     outputStream « "<file*.bgr>: each one of the files to be joined" « endl;
00037 }
```

5.10 JOIN.cpp

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Language5
00003  * Curso 2022/2023
00004  */
00005
00015 #include <iostream>
00016 #include <cstring>
00017
00018 #include "BigramCounter.h"
00019 #include "Language.h"
00020
00021 using namespace std;
00022
00028 void showEnglishHelp(ostream& outputStream) {
00029     outputStream « "Error, run with the following parameters:" « endl;
00030     outputStream « "JOIN [-t|-b] [-o <outputFile.bgr>] <file1.bgr> [<file2.bgr> ... <filen.bgr>] " «
      endl;
00031     outputStream « "       join the Language files <file1.bgr> <file2.bgr> ... into <outputFile.bgr>"
      « endl;
00032     outputStream « endl;
00033     outputStream « "Parameters:" « endl;
00034     outputStream « "-t|-b: text mode or binary mode for the output file (-t by default)" « endl;
00035     outputStream « "-o <outputFile.bgr>: name of the output file (output.bgr by default)" « endl;
00036     outputStream « "<file*.bgr>: each one of the files to be joined" « endl;
00037 }
00038
00052 int main(int argc, char* argv[]) {
00053     Language inputLanguage, inputLanguageAux;
00054     string outputFileName = "output.bgr";
00055     int nInputFiles;
00056     int firstInputFileArg = -1, // position in argv of the first input file name
00057         lastInputFileArg = -1; // position in argv of the last input file name
00058
00059     char mode = 't'; // text mode ('t') or binary ('b')
00060     bool hasBeenReadInitialParameters = false; // true if all the parameters with - has been read
00061
00062     for (int i = 1; i < argc && !hasBeenReadInitialParameters; ++i) { // Loop to process the main()
      parameters
00063         if (argv[i][0] == '-' && !hasBeenReadInitialParameters) {
00064             if(strcmp(argv[i], "-t") == 0){
00065                 mode = 't';
00066             }
00067             else if(strcmp(argv[i], "-b") == 0){
00068                 mode = 'b';
00069             }
00070             else if (strcmp(argv[i], "-o") == 0) {
00071                 if ((i + 1) < argc) { // If at least another parameter is provided (output file name)
00072                     outputFileName = argv[i + 1];
00073                     ++i;
00074                 }
00075             }
00076             else{
00077                 showEnglishHelp(cerr);
00078                 return 1;
00079             }
00080         }
00081         else {
00082             hasBeenReadInitialParameters = true;
00083             firstInputFileArg = i;
00084         }
00085     }
00086     lastInputFileArg = argc-1;
00087     if (firstInputFileArg < 0) {
00088         showEnglishHelp(cerr);
00089         return 1;
00090     }
00091
00092     inputLanguage.load(argv[firstInputFileArg]);
00093     for (int i = firstInputFileArg+1; i <= lastInputFileArg; i++) {
00094         inputLanguageAux.load(argv[i]);
00095         if(inputLanguageAux.getLanguageId() == inputLanguage.getLanguageId()){
00096             inputLanguage+=inputLanguageAux;
00097         }
00098     }
00099     inputLanguage.sort();
00100     inputLanguage.save(outputFileName.c_str(),mode);
00101     return 0;
00102 }
00103
```

5.11 src/Kmer.cpp File Reference

```
#include <string>
#include "Kmer.h"
```

Functions

- bool IsValidNucleotide (char nucleotide, const string &validNucleotides)
- void ToLower (Kmer &kmer)

  Converts to lowercase the characters (nucleotides) of the given Kmer.
- void ToUpper (Kmer &kmer)

  Converts to uppercase the characters (nucleotides) of the given Kmer.
- std::ostream & operator<< (std::ostream &os, const Kmer &kmer)

  Overloading of the stream insertion operator for Kmer class. It inserts the characters (nucleotides) of the given Kmer in the output string.
- std::istream & operator>> (std::istream &is, Kmer &kmer)

  Overloading of the stream extraction operator for Kmer class. It reads a list of characters from the input string that will set the list of nucleotides of the given Kmer.
- bool operator> (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator > for Kmer class.
- bool operator< (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator < for Kmer class.
- bool operator== (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator == for Kmer class.
- bool operator!= (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator != for Kmer class.
- bool operator<= (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator <= for Kmer class.
- bool operator>= (const Kmer &kmer1, const Kmer &kmer2)

  Overloading of the operator >= for Kmer class.

5.11.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es

Andrés Cano Utrera acu@decsai.ugr.es

Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 23 October 2023, 12:27

Definition in file Kmer.cpp.

5.11.2 Function Documentation

5.11.2.1 IsValidNucleotide()

```
bool IsValidNucleotide (
            char nucleotide,
            const string & validNucleotides )
```

Definition at line 148 of file Kmer.cpp.

```
00148                                                                    {
00149     return validNucleotides.find(nucleotide)!=string::npos;
00150 }
```

5.11.2.2 operator"!=()

```
bool operator!= (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator != for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if the two kmers contains different text; false otherwise

Definition at line 189 of file Kmer.cpp.

```
00189                                                {
00190     return !(kmer1 == kmer2);
00191 }
```

5.11.2.3 operator<()

```
bool operator< (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator < for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 < kmer2; false otherwise

Definition at line 181 of file Kmer.cpp.

```
00181                                                   {
00182     return kmer2 > kmer1;
00183 }
```

5.11.2.4 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Kmer & kmer )
```

Overloading of the stream insertion operator for Kmer class. It inserts the characters (nucleotides) of the given Kmer in the output string.

Parameters

os	The output stream to be used
kmer	the Kmer object

Returns

os A reference to the output stream

Definition at line 164 of file Kmer.cpp.

```
00164                                                   {
00165     os « kmer._text;
00166     return os;
00167 }
```

5.11.2.5 operator<=()

```
bool operator<= (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator <= for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 <= kmer2; false otherwise

Definition at line 193 of file Kmer.cpp.

```
00193                                                   {
00194     return !(kmer1 > kmer2);
00195 }
```

5.11.2.6 operator==()

```
bool operator== (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator == for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if the two kmers contains the same text; false otherwise

Definition at line 185 of file Kmer.cpp.

```
00185                                                   {
00186     return !(kmer1<kmer2 || kmer1>kmer2);
00187 }
```

5.11.2.7 operator>()

```
bool operator> (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator > for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 > kmer2; false otherwise

Definition at line 177 of file Kmer.cpp.

```
00177                                                   {
00178     return kmer1.toString() > kmer2.toString();
00179 }
```

5.11.2.8 operator>=()

```
bool operator>= (
            const Kmer & kmer1,
            const Kmer & kmer2 )
```

Overloading of the operator >= for Kmer class.

Parameters

kmer1	a Kmer object
kmer2	a Kmer object

Returns

true if kmer1 >= kmer2; false otherwise

Definition at line 197 of file Kmer.cpp.

```
00197                                                   {
00198     return !(kmer1 < kmer2);
00199 }
```

5.11.2.9 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            Kmer & kmer )
```

Overloading of the stream extraction operator for Kmer class. It reads a list of characters from the input string that will set the list of nucleotides of the given Kmer.

Parameters

is	The input stream to be used
kmer	the Kmer object

Returns

`is` the input stream

Definition at line 169 of file Kmer.cpp.

```
00169                                                   {
00170     string chain;
```

```
00171     is » chain;
00172     kmer = Kmer(chain);
00173
00174     return is;
00175 }
```

5.11.2.10 ToLower()

```
void ToLower (
              Kmer & kmer )
```

Converts to lowercase the characters (nucleotides) of the given Kmer.

Deprecated This function could go away in future versions

Parameters

kmer	A Kmer object

Definition at line 152 of file Kmer.cpp.

```
00152                           {
00153     for(int i=0; i<kmer.size(); i++){
00154        kmer.at(i) = tolower(kmer.at(i));
00155     }
00156 }
```

5.11.2.11 ToUpper()

```
void ToUpper (
              Kmer & kmer )
```

Converts to uppercase the characters (nucleotides) of the given Kmer.

Deprecated This function could go away in future versions

Parameters

kmer	A Kmer object

Definition at line 158 of file Kmer.cpp.

```
00158                           {
00159     for(int i=0; i<kmer.size(); i++){
00160        kmer.at(i) = toupper(kmer.at(i));
00161     }
00162 }
```

5.12 Kmer.cpp

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00015 #include <string>
00016
00017
00018 #include "Kmer.h"
00019
00020 using namespace std;
00021
00022 Kmer::Kmer(int k){
00023     if(k<1){
00024         throw std::invalid_argument(string("Kmer(int k): ") +
00025                 "invalid length " + to_string(k));
00026     }
00027     this->_text = string(k, MISSING_NUCLEOTIDE);
00028 }
00029
00030 Kmer::Kmer(const std::string& text) {
00031     if(text.size()==0){
00032         throw std::invalid_argument(string("Kmer(const std::string& text): ") +
00033                 "text is an empty string ");
00034     }
00035     this->_text = string(text);
00036 }
00037
00038 int Kmer::getK() const {
00039     return this->size();
00040 }
00041
00042 int Kmer::size() const {
00043     return this->_text.size();
00044 }
00045
00046 std::string Kmer::toString() const {
00047     return _text;
00048 }
00049
00050 const char& Kmer::at(int index) const{
00051     if(index<0 || index>=this->size()){
00052         throw std::out_of_range(string("const char& Kmer::at(int index) const: ") +
00053                 "invalid position " + to_string(index));
00054     }
00055     else{
00056         return _text[index];
00057     }
00058 }
00059
00060 char& Kmer::at(int index){
00061     if(index<0 || index>=this->size()){
00062         throw std::out_of_range(string("char& Kmer::at(int index): ") +
00063                 "invalid position " + to_string(index));
00064     }
00065     else{
00066         return _text[index];
00067     }
00068 }
00069
00070 void Kmer::toLower() {
00071 //    ::ToLower(*this);
00072     for(int i=0; i<size(); i++){
00073        at(i) = tolower(at(i));
00074     }
00075 }
00076
00077 void Kmer::toUpper() {
00078 //    ::ToUpper(*this);
00079     for(int i=0; i<size(); i++){
00080        at(i) = toupper(at(i));
00081     }
00082 }
00083
00084 void Kmer::normalize(const string& validNucleotides){
00085 //    Version that does not use *this
00086 //    Kmer aux(this->toString());
00087 //    toLower(aux);
00088 //    for(int i=0; i<aux.size(); i++){
00089 //        if(!isValidNucleotide(aux.at(i), validNucleotides)){
00090 //            this->at(i) = Kmer.UNKNOWN_NUCLEOTIDE;
00091 //        }
```

```
00092 //        else{
00093 //            this->at(i) = aux.at(i);
00094 //        }
00095 //    }
00096
00097 //    ::ToUpper(*this);
00098     this->toUpper();
00099     for(int i=0; i<this->size(); i++){
00100         if(!IsValidNucleotide(this->at(i), validNucleotides)){
00101             this->at(i) = Kmer::MISSING_NUCLEOTIDE;
00102         }
00103     }
00104 }
00105
00106 Kmer Kmer::complementary(const string& nucleotides,
00107          const string& complementaryNucleotides) const{
00108
00109     if(nucleotides.size() != complementaryNucleotides.size()){
00110         throw std::invalid_argument(
00111             string("Kmer Kmer::complementary(const string& nucleotides, ") +
00112                 "const string& complementaryNucleotides) const:" +
00113                 " nucleotides and complementaryNucleotides have different lengths.");
00114     }
00115
00116     int pos;
00117     Kmer result(*this);
00118
00119     for(int i=0; i<result.size(); i++){
00120         pos =  nucleotides.find(result.at(i));
00121         if(pos !=string::npos ){ // if found
00122             result.at(i) = complementaryNucleotides.at(pos);
00123         }
00124     }
00125     return result;
00126 }
00127
00128 void Kmer::write(ostream& outputStream) const {
00129     outputStream.write(_text.c_str(),  _text.size()+1 );
00130 }
00131
00132 void Kmer::read(istream& inputStream) {
00133     char* aux=new char[_text.size()+1];
00134
00135     inputStream.read(aux, _text.size()+1 );
00136     _text = aux;
00137     delete[] aux;
00138 }
00139
00140 const char& Kmer::operator[](int index) const {
00141     return _text[index];
00142 }
00143
00144 char& Kmer::operator[](int index) {
00145     return _text[index];
00146 }
00147
00148 bool IsValidNucleotide(char nucleotide, const string& validNucleotides) {
00149     return validNucleotides.find(nucleotide)!=string::npos;
00150 }
00151
00152 void ToLower(Kmer& kmer) {
00153     for(int i=0; i<kmer.size(); i++){
00154        kmer.at(i) = tolower(kmer.at(i));
00155     }
00156 }
00157
00158 void ToUpper(Kmer& kmer) {
00159     for(int i=0; i<kmer.size(); i++){
00160        kmer.at(i) = toupper(kmer.at(i));
00161     }
00162 }
00163
00164 std::ostream& operator«(std::ostream& os, const Kmer& kmer) {
00165     os « kmer._text;
00166     return os;
00167 }
00168
00169 std::istream& operator»(std::istream& is, Kmer& kmer) {
00170     string chain;
00171     is » chain;
00172     kmer = Kmer(chain);
00173
00174     return is;
00175 }
00176
00177 bool operator>(const Kmer& kmer1, const Kmer& kmer2){
00178     return kmer1.toString() > kmer2.toString();
```

```
00179 }
00180
00181 bool operator<(const Kmer& kmer1, const Kmer& kmer2){
00182     return kmer2 > kmer1;
00183 }
00184
00185 bool operator==(const Kmer& kmer1, const Kmer& kmer2){
00186     return !(kmer1<kmer2 || kmer1>kmer2);
00187 }
00188
00189 bool operator!=(const Kmer& kmer1, const Kmer& kmer2){
00190     return !(kmer1 == kmer2);
00191 }
00192
00193 bool operator<=(const Kmer& kmer1, const Kmer& kmer2){
00194     return !(kmer1 > kmer2);
00195 }
00196
00197 bool operator>=(const Kmer& kmer1, const Kmer& kmer2){
00198     return !(kmer1 < kmer2);
00199 }
```

5.13 src/KmerCounter.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <cmath>
#include "KmerCounter.h"
#include "Kmer.h"
```

5.13.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es

Andrés Cano Utrera acu@decsai.ugr.es

Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 7 November 2023, 14:00

Definition in file KmerCounter.cpp.

5.14 KmerCounter.cpp

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00015 #include <iostream>
00016 #include <fstream>
00017 #include <string>
00018 #include <cstring>
00019 #include <cmath>
00020
00021 #include "KmerCounter.h"
00022 #include "Kmer.h"
00023
```

```
00024 using namespace std;
00025
00034 const char* const KmerCounter::DEFAULT_VALID_NUCLEOTIDES="ACGT";
00035
00036
00037 //char readNextValidChar(istream& fe, const string& validChars);
00038
00039 KmerCounter::KmerCounter(int k, const string& validNucleotides) {
00040     _validNucleotides = validNucleotides;
00041     _allNucleotides = string(1, Kmer::MISSING_NUCLEOTIDE) + _validNucleotides;
00042     _k = k;
00043     allocate(getNumRows(), getNumCols());
00044     initFrequencies();
00045 }
00046
00047 KmerCounter::KmerCounter(const KmerCounter& orig) {
00048     allocate(orig.getNumRows(), orig.getNumCols());
00049     copy(orig);
00050 }
00051
00052 KmerCounter::~KmerCounter() {
00053     deallocate();
00054 }
00055
00056 int KmerCounter::getNumNucleotides() const {
00057     return _allNucleotides.size();
00058 }
00059
00060 int KmerCounter::getK() const {
00061     return _k;
00062 }
00063
00064 int KmerCounter::getNumKmers() const{
00065    return pow(getNumNucleotides(),_k);
00066 }
00067
00068 int KmerCounter::getNumberActiveKmers() const {
00069     int counter = 0;
00070     int nRows= getNumRows();
00071     int nCols = getNumCols();
00072
00073     for (int row = 0; row < nRows; ++row) {
00074         for (int column = 0; column < nCols; ++column) {
00075             if (_frequency[row][column] > 0)
00076                 counter++;
00077         }
00078     }
00079     return counter;
00080 }
00081
00082 //bool KmerCounter::setFrequency(const Kmer& kmer, int frequency) {
00083 //    int row, column;
00084 //
00085 //    this->getRowColumn(row, column);
00086 //    if (0 <= row && row < getNumNucleotides() && 0 <= column && column < getNumNucleotides()) {
00087 //        _frequency[row][column] = frequency;
00088 //        return true;
00089 //    }
00090 //
00091 //    return false;
00092 //}
00093
00094 void KmerCounter::increaseFrequency(const Kmer& kmer, int frequency) {
00095     int row, column;
00096
00097     this->getRowColumn(kmer, row, column);
00098     if(row<0 || column <0){
00099         throw std::invalid_argument(
00100                 string("void KmerCounter::increaseFrequency(const Kmer& kmer, int frequency): ") +
00101                 kmer.toString() + " contains invalid nucleotides");
00102     }
00103     else{
00104          _frequency[row][column] += frequency;
00105     }
00106 }
00107
00108 KmerCounter& KmerCounter::operator=(const KmerCounter& orig) {
00109     if (this != &orig) {
00110         deallocate();
00111         allocate(orig.getNumRows(), orig.getNumCols());
00112         copy(orig);
00113     }
00114     return *this;
00115 }
00116
00117 KmerCounter& KmerCounter::operator+=(const KmerCounter& rkc) {
00118     if (_allNucleotides == rkc._allNucleotides && _k==rkc._k) {
```

```
00119         for (int f = 0; f < getNumRows(); f++)
00120             for (int c = 0; c < getNumCols(); c++)
00121                 _frequency[f][c] += rkc._frequency[f][c];
00122     }
00123     else{
00124         throw std::invalid_argument(
00125                 string("KmerCounter& KmerCounter::operator+=(const KmerCounter& rkc): ") +
00126                 " the given argument contains a different set nucleotides or a different K");
00127     }
00128     return *this;
00129 }
00130
00131 const int& KmerCounter::operator()(int row, int column) const {
00132     return _frequency[row][column];
00133 }
00134
00135 int& KmerCounter::operator()(int row, int column) {
00136     return _frequency[row][column];
00137 }
00138
00139 void KmerCounter::calculateFrequencies(const char* fileName) {
00140     ifstream fe;
00141
00142     fe.open(fileName, ifstream::in);
00143     if (!fe) {
00144         throw std::ios_base::failure(
00145             string("void KmerCounter::calculateFrequencies(const char* fileName): ") +
00146                 "Error, opening file " + string(fileName));
00147     } else {
00148         int index = 0;
00149         int size;
00150         string inputString;
00151
00152         fe » inputString; // Read the DNA sequence
00153         size = inputString.size();
00154
00155         // Obtain the kmers
00156         while (size - index >= _k) { // exit if there are not enough characters to build a Kmer
00157             Kmer kmer = Kmer(inputString.substr(index, _k));
00158             kmer.normalize(this->_validNucleotides);
00159 //            cout « "KMER encontrado: " « kmer « endl;
00160             increaseFrequency(kmer);
00161             index++;
00162         }
00163         fe.close();
00164     }
00165 }
00166
00167 Profile KmerCounter::toProfile() const {
00168     int frequency;
00169     KmerFreq kmerFreq;
00170     Profile profile(this->getNumberActiveKmers());
00171 //    cout « "getNumberActiveKmers(): " « getNumberActiveKmers() « endl;
00172
00173     int nRows = this->getNumRows();
00174     int nCols = this->getNumCols();
00175     for (int pos = 0, row = 0; row < nRows; row++) {
00176         for (int col = 0; col < nCols; col++) {
00177             frequency = (*this)(row, col); // or this->_frequency[row][col];
00178             if (frequency > 0) {
00179                 Kmer kmer = this->getKmer(row, col);
00180
00181                 kmerFreq.setKmer(kmer);
00182                 kmerFreq.setFrequency(frequency);
00183 //                cout « "Kmer a insertar en Profile " « kmer « endl;
00184 //                cout « "profile size: " « profile.getSize() « endl;
00185                 profile.at(pos) = kmerFreq; // or profile[pos] = kmerFreq;
00186                 pos++;
00187             }
00188         }
00189     }
00190 //    profile.sort();
00191     return profile;
00192 }
00193
00194 //void KmerCounter::fromProfile(const Profile & profile) {
00195 //    for (int i = 0; i < profile.getSize(); i++){
00196 //        this->setFrequency(profile.at(i).getKmer(), // or
      this->increaseFrequency(profile[i].getKmer(),
00197 //            profile.at(i).getFrequency()); // or profile[i].getFrequency());
00198 //    }
00199 //}
00200
00202
00203 void KmerCounter::allocate(int nRows, int nCols) {
00204     if (nRows > 0 && nCols >0) {
00205         _frequency = new int*[nRows];
```

```
00206         _frequency[0] = new int[nRows * nCols];
00207         for (int i = 1; i < nRows; ++i)
00208             _frequency[i] = _frequency[i - 1] + nCols;
00209     }
00210     else{
00211         _frequency = nullptr;
00212     }
00213 }
00214
00215 int KmerCounter::getNumRows() const {
00216     return pow(_allNucleotides.size(), (_k + 1)/2);
00217 }
00218
00219 int KmerCounter::getNumCols() const {
00220     return pow(_allNucleotides.size(), _k - (_k + 1)/2 );
00221 }
00222
00223
00224
00225 int KmerCounter::getIndex(const std::string& kmer) const{
00226     int index = 0;
00227     int base = 1;
00228
00229     for (int i = 0; i < kmer.size(); i++) {
00230         int pos = _allNucleotides.find(kmer[kmer.size()-i-1]);
00231         if (pos < 0) // ToDo lanzar excepción ?
00232             return -1;
00233             //pos = 0;
00234         index += pos * base;
00235         base *= _allNucleotides.size();
00236     }
00237     return index;
00238 }
00239
00240 string KmerCounter::getInvertedIndex(int index, int nCharacters) const {
00241     string result(nCharacters, Kmer::MISSING_NUCLEOTIDE);
00242
00243     for (int i = result.size(); i > 0; i--) {
00244         result[i - 1] = _allNucleotides[index % _allNucleotides.size()];
00245         index = index / _allNucleotides.size();
00246     }
00247     return result;
00248 }
00249
00250 void KmerCounter::getRowColumn(const Kmer &kmer, int& row, int& column) const {
00251     int lenRowString=(_k + 1)/2;
00252     string rowString=kmer.toString().substr(0, lenRowString);
00253     string colString=kmer.toString().substr(lenRowString, _k-lenRowString);
00254
00255     row = getIndex(rowString);
00256     column = getIndex(colString);
00257 }
00258
00259 Kmer KmerCounter::getKmer(int row, int column) const {
00260     if(row<0 || column <0 || row >= this->getNumRows() || column >= this->getNumCols()){
00261         throw std::invalid_argument(
00262                 string("Kmer KmerCounter::getKmer(int row, int column) const: ") +
00263                 " invalid row or column, row = " + std::to_string(row) +
00264                 ", column = " + std::to_string(column));
00265     }
00266     string dna = this->getInvertedIndex(row, (_k + 1)/2 ) +
00267             this->getInvertedIndex(column, _k - (_k + 1)/2);
00268     return Kmer(dna);
00269 }
00270
00271 void KmerCounter::initFrequencies() {
00272     for (int row = 0; row < getNumRows(); ++row) {
00273         for (int column = 0; column < getNumCols(); ++column) {
00274             _frequency[row][column] = 0;
00275         }
00276     }
00277 }
00278
00279 void KmerCounter::deallocate() {
00280     if (_frequency != nullptr) {
00281         delete[] _frequency[0];
00282         delete[] _frequency;
00283         _frequency = nullptr;
00284     }
00285 }
00286
00287 void KmerCounter::copy(const KmerCounter& orig) {
00288     _validNucleotides = orig._validNucleotides;
00289     _allNucleotides = orig._allNucleotides;
00290     _k = orig._k;
00291
00292     for (int row = 0; row < getNumRows(); ++row) {
```

```
00293         for (int col = 0; col < getNumCols(); ++col) {
00294             (*this)(row,col) = orig(row,col); // or _frequency[row][col] = orig._frequency[row][col];
00295         }
00296     }
00297 }
00298
00299 //char readNextValidChar(istream& inputStream, const string& validChars) {
00300 //    // TODO Ineficiente. Podría mejorarse usando un map como hace Huete

00301 //    // suponiendo que los caracteres en cadena están ordenados por orden alfabético
00302 //    char character, result = '\0';
00303 //    bool stop = false;
00304 //    while (inputStream && !stop) {
00305 //        character = tolower(inputStream.get());
00306 //        if (inputStream) {
00307 //            if (validChars.find(character) < validChars.size()) {
00308 //                result = character;
00309 //                stop = true;
00310 //            }
00311 //        }
00312 //    }
00313 //    return result;
00314 //}
00315
```

5.15 src/KmerFreq.cpp File Reference

```
#include <string>
#include "KmerFreq.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const KmerFreq &kmerFreq)

  Overloading of the stream insertion operator for KmerFreq class.
- std::istream & operator>> (std::istream &is, KmerFreq &kmerFreq)

  Overloading of the stream extraction operator for KmerFreq class.
- bool operator> (const KmerFreq &kmerFreq1, const KmerFreq &kmerFrec2)

  Overloading of the relational operator > for KmerFreq class.
- bool operator< (const KmerFreq &kmerFreq1, const KmerFreq &kmerFreq2)

  Overloading of the operator < for KmerFreq class.
- bool operator== (const KmerFreq &kmerFreq1, const KmerFreq &kmerFrec2)

  Overloading of the operator == for Kmer class.
- bool operator!= (const KmerFreq &kmerFreq1, const KmerFreq &kmerFreq2)

  Overloading of the operator != for KmerFreq class.
- bool operator<= (const KmerFreq &kmerFreq1, const KmerFreq &kmerFreq2)

  Overloading of the operator <= for KmerFreq class.
- bool operator>= (const KmerFreq &kmerFreq1, const KmerFreq &kmerFreq2)

  Overloading of the operator >= for KmerFreq class.

5.15.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es
Javier Martínez Baena jbaena@ugr.es

Created on 26 October 2023, 11:22

Definition in file KmerFreq.cpp.

5.15.2 Function Documentation

5.15.2.1 operator"!=()

```
bool operator!= (
            const KmerFreq & kmerFreq1,
            const KmerFreq & kmerFreq2 )
```

Overloading of the operator != for KmerFreq class.

Parameters

kmerFreq1	a Kmer object
kmerFreq2	a Kmer object

Returns

true if the two kmerFreq1 are not equals (see operator==); false otherwise

Definition at line 76 of file KmerFreq.cpp.

```
00076                                                          {
00077     return !(kmerFreq1 == kmerFreq2);
00078 }
```

5.15.2.2 operator<()

```
bool operator< (
            const KmerFreq & kmerFreq1,
            const KmerFreq & kmerFreq2 )
```

Overloading of the operator < for KmerFreq class.

Parameters

kmerFreq1	a Kmer object
kmerFreq2	a Kmer object

Returns

true if kmerFreq1 < kmerFreq2; false otherwise

Definition at line 68 of file KmerFreq.cpp.

```
00068                                                          {
00069     return kmerFreq2 > kmerFreq1;
00070 }
```

5.15.2.3 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const KmerFreq & kmerFreq )
```

Overloading of the stream insertion operator for KmerFreq class.

Parameters

os	The output stream to be used
kmerFreq	the KmerFreq object

Returns

os A reference to the output stream

Definition at line 50 of file KmerFreq.cpp.

```
00050                                                                    {
00051     os « kmerFreq.getKmer() « " " « kmerFreq.getFrequency();
00052     return os;
00053 }
```

5.15.2.4 operator<=()

```
bool operator<= (
            const KmerFreq & kmerFreq1,
            const KmerFreq & kmerFreq2 )
```

Overloading of the operator <= for KmerFreq class.

Parameters

kmerFreq1	a Kmer object
kmerFreq2	a Kmer object

Returns

true if kmerFreq1 <= kmerFreq2; false otherwise

Definition at line 80 of file KmerFreq.cpp.

```
00080                                                                    {
00081     return !(kmerFreq1 > kmerFreq2);
00082 }
```

5.15.2.5 operator==()

```
bool operator== (
            const KmerFreq & kmerFreq1,
            const KmerFreq & kmerFreq2 )
```

Overloading of the operator == for Kmer class.

Parameters

kmerFreq1	a KmerFreq object
kmerFreq2	a KmerFreq object

Returns

true if the two kmers contains the same pair Kmer-frequency; false otherwise

Definition at line 72 of file KmerFreq.cpp.

```
00072                                                                    {
00073     return !(kmerFreq1<kmerFrec2 || kmerFreq1>kmerFrec2);
00074 }
```

5.15.2.6 operator>()

```
bool operator> (
            const KmerFreq & kmerFreq1,
            const KmerFreq & kmerFreq2 )
```

Overloading of the relational operator > for KmerFreq class.

Parameters

kmerFreq1	The first object to be compared
kmerFreq2	The second object to be compared

Returns

true if the frequency of `kmerFreq1` is greater than that of `kmerFreq2` or if both frequencies are equals and the text of `kmerFreq1` is minor than the text of `kmerFreq2`; false otherwise

Definition at line 62 of file KmerFreq.cpp.

```
00062                                                                    {
00063     return (kmerFreq1.getFrequency() > kmerFrec2.getFrequency()) ||
00064             ((kmerFreq1.getFrequency() == kmerFrec2.getFrequency()) &&
00065             (kmerFreq1.getKmer() < kmerFrec2.getKmer()));
00066 }
```

5.15.2.7 operator>=()

```
bool operator>= (
            const KmerFreq & kmerFreq1,
            const KmerFreq & kmerFreq2 )
```

Overloading of the operator >= for KmerFreq class.

Parameters

kmerFreq1	a Kmer object
kmerFreq2	a Kmer object

Returns

true if kmerFreq1 >= kmerFreq2; false otherwise

Definition at line 84 of file KmerFreq.cpp.

```
00084                                                                   {
00085     return !(kmerFreq1 < kmerFreq2);
00086 }
```

5.15.2.8 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            KmerFreq & kmerFreq )
```

Overloading of the stream extraction operator for KmerFreq class.

Parameters

is	The input stream to be used
kmerFreq	the KmerFreq object

Returns

`is` A reference to the input stream

Definition at line 55 of file KmerFreq.cpp.

```
00055                                                                   {
00056     is » kmerFreq._kmer ;
00057     is » kmerFreq._frequency;
00058
00059     return is;
00060 }
```

5.16 KmerFreq.cpp

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00016 #include <string>
00017
00018 #include "KmerFreq.h"
00019
00020 using namespace std;
00021
00022
```

```
00023 KmerFreq::KmerFreq():_kmer(string(1, Kmer::MISSING_NUCLEOTIDE)), _frequency(0) {
00024 }
00025
00026 const Kmer& KmerFreq::getKmer() const {
00027     return _kmer;
00028 }
00029
00030 int KmerFreq::getFrequency() const {
00031     return _frequency;
00032 }
00033
00034 void KmerFreq::setKmer(const Kmer& kmer) {
00035     this->_kmer = kmer;
00036 }
00037
00038 void KmerFreq::setFrequency(int frequency) {
00039     if(frequency<0){
00040         throw  std::out_of_range(string("void KmerFreq::setFrequency(int frequency): ") +
00041                 "invalid frequency " + to_string(frequency));
00042     }
00043     this->_frequency = frequency;
00044 }
00045
00046 string KmerFreq::toString() const {
00047     return _kmer.toString() + " " + to_string(_frequency);
00048 }
00049
00050 std::ostream& operator«(std::ostream& os, const KmerFreq& kmerFreq) {
00051     os « kmerFreq.getKmer() « " " « kmerFreq.getFrequency();
00052     return os;
00053 }
00054
00055 std::istream& operator»(std::istream& is, KmerFreq& kmerFreq) {
00056     is » kmerFreq._kmer ;
00057     is » kmerFreq._frequency;
00058
00059     return is;
00060 }
00061
00062 bool operator>(const KmerFreq &kmerFreq1, const KmerFreq &kmerFrec2) {
00063     return (kmerFreq1.getFrequency() > kmerFrec2.getFrequency()) ||
00064             ((kmerFreq1.getFrequency() == kmerFrec2.getFrequency()) &&
00065             (kmerFreq1.getKmer() < kmerFrec2.getKmer()));
00066 }
00067
00068 bool operator<(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2){
00069     return kmerFreq2 > kmerFreq1;
00070 }
00071
00072 bool operator==(const KmerFreq &kmerFreq1, const KmerFreq &kmerFrec2) {
00073     return !(kmerFreq1<kmerFrec2 || kmerFreq1>kmerFrec2);
00074 }
00075
00076 bool operator!=(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2){
00077     return !(kmerFreq1 == kmerFreq2);
00078 }
00079
00080 bool operator<=(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2){
00081     return !(kmerFreq1 > kmerFreq2);
00082 }
00083
00084 bool operator>=(const KmerFreq& kmerFreq1, const KmerFreq& kmerFreq2){
00085     return !(kmerFreq1 < kmerFreq2);
00086 }
00087
00088 void KmerFreq::write(ostream& outputStream) const {
00089     _kmer.write(outputStream);
00090     outputStream.write(reinterpret_cast<const char *> (&_frequency), sizeof (int));
00091 }
00092
00093 void KmerFreq::read(istream& inputStream) {
00094     _kmer.read(inputStream);
00095     inputStream.read(reinterpret_cast<char *> (&_frequency), sizeof(int));
00096 }
```

5.17 src/LEARN.cpp File Reference

```
#include <iostream>
#include <cstring>
#include "KmerCounter.h"
```

```
#include "Profile.h"
```

Functions

- void showSpanishHelp (ostream &outputStream)
- void showEnglishHelp (ostream &outputStream)
- int main (int argc, char ∗argv[ ])

5.17.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 9 November 2023, 13:30

Definition in file LEARN.cpp.

5.17.2 Function Documentation

5.17.2.1 main()

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

This program learns a Profile model from a set of input DNA files (file1.dna, file2.dna, ...). The learned Profile object is then ordered by frequency and saved in the file outputFilename (or output.prf if the output file is not provided). Running example:

LEARN [-t|-b] [-k kValue] [-i profileId] [-o outputFilename] <file1.dna> [<file2.dna> <file3.dna> ....]

Parameters

argc	The number of command line parameters
argv	The vector of command line parameters (cstrings)

Returns

0 If there is no error; a value > 0 if error

Definition at line 70 of file LEARN.cpp.

```
00070                                {
00071     string validNucleotides = "ACGT";
00072     int profileArg = -1,
00073             outputFileArg = -1, // position in argv of the name of the output file name
00074             firstInputFileArg = -1, // position in argv of the first input file name
00075             lastInputFileArg = -1; // position in argv of the last input file name
00076     int kValue=5;
00077     char mode = 't'; // text mode ('t') or binary ('b')
00078     bool hasBeenReadInitialParameters = false; // true if all the parameters with - has been read
00079     string profileId = "unknown";
00080
00081
00082     for (int i = 1; i < argc && !hasBeenReadInitialParameters; ++i) { // Loop to process the main()
      parameters
00083         if (argv[i][0] == '-' && !hasBeenReadInitialParameters) {
00084             if(strcmp(argv[i], "-t") == 0){
00085                 mode = 't';
00086             }
00087             else if(strcmp(argv[i], "-k") == 0){
00088                 if ((i + 1) < argc) { // If at least another parameter is provided (K value)
00089                     kValue = atoi(argv[i+1]);
00090                     ++i;
00091                 }
00092             }
00093             else if (strcmp(argv[i], "-n") == 0) {
00094                 if ((i + 1) < argc) { // If at least another parameter is provided (profile
      identifier)
00095                     validNucleotides = argv[i+1];
00096                     ++i;
00097                 }
00098             }
00099             else if(strcmp(argv[i], "-b") == 0){
00100                 mode = 'b';
00101             }
00102             else if (strcmp(argv[i], "-i") == 0) {
00103                 if ((i + 1) < argc) { // If at least another parameter is provided (profile
      identifier)
00104                     profileArg = i + 1;
00105                     ++i;
00106                 }
00107             }
00108             else if (strcmp(argv[i], "-o") == 0) {
00109                 if ((i + 1) < argc) { // If at least another parameter is provided (output file name)
00110                     outputFileArg = i + 1;
00111                     ++i;
00112                 }
00113             }
00114             else{
00115                 showEnglishHelp(cerr);
00116                 return 1;
00117             }
00118         } else {
00119             hasBeenReadInitialParameters = true;
00120             firstInputFileArg = i;
00121         }
00122     }
00123     lastInputFileArg = argc-1;
00124
00125     if (firstInputFileArg < 0) {
00126         showEnglishHelp(cerr);
00127         return 1;
00128     } else {
00129         KmerCounter kmerCounter(kValue, validNucleotides);
00130         Profile profile;
00131
00132         if (profileArg > 0) {
00133             profileId = argv[profileArg];
00134         }
00135         for (int i = firstInputFileArg; i <= lastInputFileArg; i++) {
00136             KmerCounter kmerCounterAux(kValue, validNucleotides);
00137
00138             kmerCounterAux.calculateFrequencies(argv[i]);
00139             kmerCounter +=kmerCounterAux;
00140         }
00141
00142         profile = kmerCounter.toProfile();
00143         profile.setProfileId(profileId);
00144         profile.sort();
00145         if (outputFileArg < 0) {
00146             profile.save("output.prf", mode);
00147         } else {
00148             profile.save(argv[outputFileArg], mode);
00149         }
00150     }
00151     return 0;
00152 }
```

5.17.2.2 showEnglishHelp()

```
void showEnglishHelp (
            ostream & outputStream )
```

Shows help about the use of this program in the given output stream

Parameters

outputStream	The output stream where the help will be shown (for example, cout, cerr, etc)

Definition at line 44 of file LEARN.cpp.

```
00044                                          {
00045     outputStream « "Error, run with the following parameters:" « endl;
00046     outputStream « "LEARN [-t|-b] [-k kValue]  [-i profileId] [-o outputFilename] <file1.dna>
      [<file2.dna> <file3.dna> .... ]" « endl;
00047     outputStream « "          learn the model for the profile profileId from the DNA files
      <file1.dna> <file2.dna> <file3.dna> ..." « endl;
00048     outputStream « endl;
00049     outputStream « "Parameters:" « endl;
00050     outputStream « "-t|-b: text mode or binary mode for the output file (-t by default)" « endl;
00051     outputStream « "-k kValue: number of nucleotides in a kmer (5 by default)" « endl;
00052     outputStream « "-n nucletidesSet: set of possible nucleotides in a kmer (ACGT by default). "
00053                  « "Note that the characters should be provided in uppercase" « endl;
00054     outputStream « "-i profileId: profile identifier (unknown by default)" « endl;
00055     outputStream « "-o outputFilename: name of the output file (output.prf by default)" « endl;
00056     outputStream « "<file1.dna> <file2.dna> <file3.dna> ....: names of the input files (at least one
      is mandatory)" « endl;
00057 }
```

5.17.2.3 showSpanishHelp()

```
void showSpanishHelp (
            ostream & outputStream )
```

Definition at line 24 of file LEARN.cpp.

```
00024                                          {
00025     outputStream « "Error, ejecute con los siguientes parámetros:" « endl;
00026     outputStream « "LEARN [-t|-b] [-k kValor] [-i profileId] [-o ficheroSalida] fichero1.dna
      fichero2.dna fichero3.dna .... " « endl;
00027     outputStream « "          aprende el modelo para el perfil profileId a partir de los ficheros DNA
      <fichero1.dna> <fichero2.dna> <fichero3.dna> ..." « endl;
00028     outputStream « endl;
00029     outputStream « "Parámetros:" « endl;
00030     outputStream « "-t|-b: modo texto o modo binario para el fichero de salida (-t por defecto)" «
      endl;
00031     outputStream « "-k kValor: número de nucleótidos en un kmer (5 por defecto)" « endl;
00032     outputStream « "-n nucletidos: conjunto de posibles nucleótidos en un kmer (ACGT por defecto). "
00033                  « "Nótese que los caracteres debería darse en mayúscula" « endl;
00034     outputStream « "-i perfilId: identificador del perfil (unknown por defecto)" « endl;
00035     outputStream « "-o ficheroSalida:  nombre del fichero de salida (output.bgr por defecto)" « endl;
00036     outputStream « "texto1.txt texto2.txt texto3.txt ....: nombres de los ficheros de entrada (debe
      haber al menos 1)" « endl;
00037 }
```

5.18 LEARN.cpp

Go to the documentation of this file.

```
00001 /*
```

```
00002  * Metodología de la Programación: Profile5
00003  * Curso 2023/2024
00004  */
00005
00015 #include <iostream>
00016 #include <cstring>
00017
00018 #include "KmerCounter.h"
00019 #include "Profile.h"
00020
00021 using namespace std;
00022
00023
00024 void showSpanishHelp(ostream& outputStream) {
00025     outputStream « "Error, ejecute con los siguientes parámetros:" « endl;
00026     outputStream « "LEARN [-t|-b] [-k kValor] [-i profileId] [-o ficheroSalida] fichero1.dna
      fichero2.dna fichero3.dna .... " « endl;
00027     outputStream « "           aprende el modelo para el perfil profileId a partir de los ficheros DNA
      <fichero1.dna> <fichero2.dna> <fichero3.dna> ..." « endl;
00028     outputStream « endl;
00029     outputStream « "Parámetros:" « endl;
00030     outputStream « "-t|-b: modo texto o modo binario para el fichero de salida (-t por defecto)" «
      endl;
00031     outputStream « "-k kValor: número de nucleótidos en un kmer (5 por defecto)" « endl;
00032     outputStream « "-n nucletidos: conjunto de posibles nucleótidos en un kmer (ACGT por defecto). "
00033                  « "Nótese que los caracteres debería darse en mayúscula" « endl;
00034     outputStream « "-i perfilId: identificador del perfil (unknown por defecto)" « endl;
00035     outputStream « "-o ficheroSalida:  nombre del fichero de salida (output.bgr por defecto)" « endl;
00036     outputStream « "texto1.txt texto2.txt texto3.txt ....: nombres de los ficheros de entrada (debe
      haber al menos 1)" « endl;
00037 }
00038
00044 void showEnglishHelp(ostream& outputStream) {
00045     outputStream « "Error, run with the following parameters:" « endl;
00046     outputStream « "LEARN [-t|-b] [-k kValue]  [-i profileId] [-o outputFilename] <file1.dna>
      [<file2.dna> <file3.dna> .... ]" « endl;
00047     outputStream « "           learn the model for the profile profileId from the DNA files
      <file1.dna> <file2.dna> <file3.dna> ..." « endl;
00048     outputStream « endl;
00049     outputStream « "Parameters:" « endl;
00050     outputStream « "-t|-b: text mode or binary mode for the output file (-t by default)" « endl;
00051     outputStream « "-k kValue: number of nucleotides in a kmer (5 by default)" « endl;
00052     outputStream « "-n nucletidesSet: set of possible nucleotides in a kmer (ACGT by default). "
00053                  « "Note that the characters should be provided in uppercase" « endl;
00054     outputStream « "-i profileId: profile identifier (unknown by default)" « endl;
00055     outputStream « "-o outputFilename: name of the output file (output.prf by default)" « endl;
00056     outputStream « "<file1.dna> <file2.dna> <file3.dna> ....: names of the input files (at least one
      is mandatory)" « endl;
00057 }
00058
00070 int main(int argc, char *argv[]) {
00071     string validNucleotides = "ACGT";
00072     int profileArg = -1,
00073             outputFileArg = -1, // position in argv of the name of the output file name
00074             firstInputFileArg = -1, // position in argv of the first input file name
00075             lastInputFileArg = -1; // position in argv of the last input file name
00076     int kValue=5;
00077     char mode = 't'; // text mode ('t') or binary ('b')
00078     bool hasBeenReadInitialParameters = false; // true if all the parameters with - has been read
00079     string profileId = "unknown";
00080
00081
00082     for (int i = 1; i < argc && !hasBeenReadInitialParameters; ++i) { // Loop to process the main()
      parameters
00083         if (argv[i][0] == '-' && !hasBeenReadInitialParameters) {
00084             if(strcmp(argv[i], "-t") == 0){
00085                 mode = 't';
00086             }
00087             else if(strcmp(argv[i], "-k") == 0){
00088                 if ((i + 1) < argc) { // If at least another parameter is provided (K value)
00089                     kValue = atoi(argv[i+1]);
00090                     ++i;
00091                 }
00092             }
00093             else if (strcmp(argv[i], "-n") == 0) {
00094                 if ((i + 1) < argc) { // If at least another parameter is provided (profile
      identifier)
00095                     validNucleotides = argv[i+1];
00096                     ++i;
00097                 }
00098             }
00099             else if(strcmp(argv[i], "-b") == 0){
00100                 mode = 'b';
00101             }
00102             else if (strcmp(argv[i], "-i") == 0) {
00103                 if ((i + 1) < argc) { // If at least another parameter is provided (profile
      identifier)
```

```
00104                     profileArg = i + 1;
00105                     ++i;
00106                 }
00107             }
00108             else if (strcmp(argv[i], "-o") == 0) {
00109                 if ((i + 1) < argc) { // If at least another parameter is provided (output file name)
00110                     outputFileArg = i + 1;
00111                     ++i;
00112                 }
00113             }
00114             else{
00115                 showEnglishHelp(cerr);
00116                 return 1;
00117             }
00118         } else {
00119             hasBeenReadInitialParameters = true;
00120             firstInputFileArg = i;
00121         }
00122     }
00123     lastInputFileArg = argc-1;
00124 
00125     if (firstInputFileArg < 0) {
00126         showEnglishHelp(cerr);
00127         return 1;
00128     } else {
00129         KmerCounter kmerCounter(kValue, validNucleotides);
00130         Profile profile;
00131 
00132         if (profileArg > 0) {
00133             profileId = argv[profileArg];
00134         }
00135         for (int i = firstInputFileArg; i <= lastInputFileArg; i++) {
00136             KmerCounter kmerCounterAux(kValue, validNucleotides);
00137 
00138             kmerCounterAux.calculateFrequencies(argv[i]);
00139             kmerCounter +=kmerCounterAux;
00140         }
00141 
00142         profile = kmerCounter.toProfile();
00143         profile.setProfileId(profileId);
00144         profile.sort();
00145         if (outputFileArg < 0) {
00146             profile.save("output.prf", mode);
00147         } else {
00148             profile.save(argv[outputFileArg], mode);
00149         }
00150     }
00151     return 0;
00152 }
00153 
```

5.19 metamain.cpp

```
00001 #ifdef LEARN
00002     #include "LEARN.cpp"
00003 #elif CLASSIFY
00004     #include "CLASSIFY.cpp"
00005 #elif JOIN
00006     #include "JOIN.cpp"
00007 #endif
00008 
```

5.20 src/Profile.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <cstring>
#include "Profile.h"
```

Functions

- ostream & operator<< (ostream &os, const Profile &language)
- istream & operator>> (istream &is, Profile &profile)

5.20.1 Detailed Description

Author

Silvia Acid Carrillo acid@decsai.ugr.es
Andrés Cano Utrera acu@decsai.ugr.es
Luis Castillo Vidal L.Castillo@decsai.ugr.es

Created on 31 October 2023, 9:40

Definition in file Profile.cpp.

5.20.2 Function Documentation

5.20.2.1 operator<<()

```
ostream & operator<< (
            ostream & os,
            const Profile & language )
```

Definition at line 316 of file Profile.cpp.

```
00316                                                       {
00317     os « language.getProfileId() « endl;
00318     os « language.getSize() « endl;
00319     for (int i = 0; i < language.getSize(); ++i) {
00320         os «  language.at(i) « endl; //or os «  idioma[i] « endl;
00321     }
00322     return os;
00323 }
```

5.20.2.2 operator>>()

```
istream & operator>> (
            istream & is,
            Profile & profile )
```

Definition at line 325 of file Profile.cpp.

```
00325                                                 {
00326     string speciesId;
00327     int numberKmers;
00328     KmerFreq kmerFreq;
00329
00330 //    profile.deallocate();
00331
00332     is » speciesId; // Read species identifier
00333     is » numberKmers; // Read the number of kmers
00334     if (numberKmers < 0) {
00335         throw std::out_of_range(
00336                 string("istream& operator»(istream& is, Profile& profile): ") +
00337                 "the number of kmers read " + to_string(numberKmers) + " is not valid ");
00338     }
00339 //    profile.allocate(numberKmers);
00340
00341     Profile iaux(numberKmers);
00342     iaux.setProfileId(speciesId);
00343 //    for (int i = 0; i < iaux.getSize(); ++i) {
00344     for (int i = 0; i < numberKmers; ++i) {
00345         is » kmerFreq;
00346 //        iaux.at(i) = kmerFreq; // or iaux[i] = bgr;
00347         iaux.append(kmerFreq);
00348     }
00349     profile = iaux;
00350     return is;
00351 }
```

5.21 Profile.cpp

Go to the documentation of this file.

```
00001 /*
00002  * Metodología de la Programación: Kmer5
00003  * Curso 2023/2024
00004  */
00005
00015 #include <iostream>
00016 #include <fstream>
00017 #include <cmath>
00018 #include <cstring>
00019
00020 #include "Profile.h"
00021
00022 using namespace std;
00023
00024 const string Profile::MAGIC_STRING_T="MP-KMER-T-1.0";
00025 const string Profile::MAGIC_STRING_B="MP-KMER-B-1.0";
00026
00027 Profile::Profile() {
00028     _profileId="unknown";
00029     allocate(INITIAL_CAPACITY);
00030 }
00031
00032 Profile::Profile(int size)  {
00033     if(size < 0){
00034         throw std::out_of_range(string("Profile::Profile(int numberKmers): ") +
00035                 "invalid numberKmers=" + to_string(size));
00036     }
00037     _profileId="unknown";
00038     allocate(size);
00039     _size = size;
00040 }
00041
00042 Profile::Profile(const Profile& orig) {
00043     allocate(orig._capacity);
00044     copy(orig);
00045 }
00046
00047 Profile::~Profile() {
00048     deallocate();
00049 }
00050
00051 Profile& Profile::operator=(const Profile& orig) {
00052     if (this != &orig) {
00053         deallocate();
00054         allocate(orig._capacity );
00055         copy(orig);
00056     }
00057     return *this;
00058 }
00059
00060 const string& Profile::getProfileId() const {
00061     return _profileId;
00062 }
00063
00064 void Profile::setProfileId(const string& id) {
00065     _profileId = id;
00066 }
00067
00068 const KmerFreq& Profile::at(int index) const {
00069     if (0 <= index && index < getSize())
00070         return _vectorKmerFreq[index];
00071     else
00072         throw std::out_of_range(string("const KmerFreq& Profile::at(int index) const: ") +
00073                 "invalid position " + to_string(index));
00074 }
00075
00076 KmerFreq& Profile::at(int index) {
00077     if (0 <= index && index < getSize())
00078         return _vectorKmerFreq[index];
00079     else
00080         throw std::out_of_range(string("KmerFreq& Profile::at(int index): ") +
00081                 "invalid position " + to_string(index));
00082 }
00083
00084 int Profile::getSize() const {
00085     return _size;
00086 }
00087
00088 int Profile::getCapacity() const {
00089     return _capacity;
00090 }
00091
```

```
00092 double Profile::getDistance(const Profile& otherProfile) const {
00093     if(this->getSize() == 0 || otherProfile.getSize() == 0){ // CAMBIADO ESTE AÑO RESPECTO A 2022/2023
00094         throw std::invalid_argument(
00095            string("double Profile::getDistance(const Profile& otherProfile) const: ")
00096                + "*this or otherProfile do not have any kmer");
00097     }
00098     int posB;
00099     int posA = 0;
00100     double dist = 0.0;
00101
00102     for (int i = 0; i < getSize(); ++i) {
00103         posB = otherProfile.findKmer (this->at(i).getKmer()); // or posB =
      B.findKmer((*this)[i].getKmer());
00104         if (posB < 0) {
00105             posB = getSize();
00106 //            posB = otherProfile.getSize(); // CAMBIADO ESTE AÑO RESPECTO A 2022/2023
00107         }
00108
00109         dist += abs(posA - posB);
00110         posA++;
00111     }
00112     return dist / (getSize() * getSize());
00113 //    return dist / (getSize() *  otherProfile.getSize() );  // CAMBIADO ESTE AÑO RESPECTO A 2022/2023
00114 }
00115
00116 int Profile::findKmer(const Kmer& kmer) const { // const string& kmer) const  {
00117     for (int i = 0; i < getSize(); ++i) {
00118         if (kmer == this->at(i).getKmer()) { // or if (kmer == (*this)[i].getKmer()) {
00119             return i;
00120         }
00121     }
00122     return -1;
00123 }
00124
00125 std::string Profile::toString() const{
00126     string outputString = this->getProfileId() + "\n" +
00127             to_string(this->getSize()) + "\n";
00128
00129     for(int i=0; i<this->getSize(); i++){
00130         outputString += this->at(i).toString() + "\n";
00131     }
00132     return outputString;
00133 }
00134
00135 void Profile::sort()  {
00136     KmerFreq aux;
00137     int pos;
00138     for (int i=0; i<getSize(); i++)  {
00139         pos = i;
00140         for (int j=i+1; j<getSize(); j++)
00141             if (this->at(j)>this->at(pos)) // or if ((*this)[j]>(*this)[pos])
00142                 pos = j;
00143         if (pos != i)  {
00144             aux = this->at(i); // or aux = (*this)[i];
00145             this->at(i) = this->at(pos);// or (*this)[i] = (*this)[pos];
00146             this->at(pos) = aux; // or (*this)[pos] = aux;
00147         }
00148     }
00149 }
00150
00151 void Profile::save(const char *fileName, char mode) const {
00152     ofstream stream(fileName, ios::out | ios::binary);
00153
00154     if (stream) {
00155         if (mode == 't') {
00156             stream « Profile::MAGIC_STRING_T « endl;
00157             stream « *this « endl;
00158         }
00159         else{ // mode == 'b'
00160             stream « Profile::MAGIC_STRING_B « endl;
00161             stream « this->getProfileId() « endl;
00162             stream « this->getSize() « endl;
00163
00164 //            stream.write(reinterpret_cast<char *>(_vectorKmerFreq),
00165 //                    sizeof(KmerFreq)*this->getSize());
00166             for(int i=0; i<this->getSize(); i++){
00167                 _vectorKmerFreq[i].write(stream);
00168             }
00169         }
00170         if (stream)  {
00171             stream.close();
00172         }
00173         else{
00174             throw std::ios_base::failure(
00175                 string("void Profile::save(const char *fileName, char mode) const: ") +
00176                         "error writing to file " + fileName);
00177         }
```

```
00178     }
00179     else{
00180         throw std::ios_base::failure(
00181                 string("void Profile::save(const char *fileName, char mode) const: ") +
00182                         "error opening file " + fileName);
00183     }
00184 }
00185
00186 void Profile::load(const char *fileName) {
00187     ifstream inputStream;
00188     inputStream.open(fileName, ifstream::in | ios::binary);
00189     string magicString;
00190
00191     if (inputStream) {
00192         inputStream » magicString;
00193         if (magicString == Profile::MAGIC_STRING_T) { // For text files
00194             inputStream » *this;
00195         } else if (magicString == Profile::MAGIC_STRING_B){ // Fo binary files
00196             int nKmers;
00197             string speciesId;
00198
00199             inputStream » speciesId; // Read Species identifier
00200             this->setProfileId(speciesId);
00201
00202             inputStream » nKmers; // Read number of kmers
00203             if (nKmers < 0) {
00204                 throw std::out_of_range(
00205                         string("void Profile::load(const char *fileName): ") +
00206                         "invalid number of kmers=" + to_string(nKmers));
00207             }
00208             inputStream.get(); // Read '\n' character that appears after nKmers
00209
00210             deallocate();
00211 //            allocate(nKmers);
00212
00213             KmerFreq kmerFreq;
00214             for(int i=0; i<nKmers; i++){
00215 //                _vectorKmerFreq[i].read(inputStream);
00216                 kmerFreq.read(inputStream);
00217                 this->append(kmerFreq);
00218             }
00219         }
00220         else {
00221             throw std::invalid_argument(
00222                 string("void Profile::load(const char *fileName): ") +
00223                 "the found magic string " + magicString + " in file " +
00224                 fileName + " is not valid ");
00225         }
00226         if (inputStream) {
00227             inputStream.close();
00228         }
00229         else{
00230             throw std::ios_base::failure(
00231                 string("void Profile::load(const char *fileName): ") +
00232                         "error reading from file " + fileName);
00233         }
00234     }
00235     else{
00236         throw std::ios_base::failure(
00237                 string("void Profile::load(const char *fileName): ") +
00238                         "error opening file " + fileName);
00239     }
00240 }
00241
00242 void Profile::append(const KmerFreq& kmerFreq) {
00243     int pos = this->findKmer(kmerFreq.getKmer().toString());
00244     if (pos >= 0) { // If found
00245         this->at(pos).setFrequency(this->at(pos).getFrequency() +
00246                 kmerFreq.getFrequency());
00247     } else { // If not found
00248         if(this->_size == this->_capacity){ // If the vector is full
00249             this->reallocate(this->_capacity + BLOCK_SIZE);
00250         }
00251         this->_size++;
00252         this->at(this->_size-1) = kmerFreq;
00253     }
00254 }
00255
00256 void Profile::normalize(const string& validNucleotides ){
00257     Kmer kmer;
00258     for(int i=0; i<_size; i++){
00259         kmer = _vectorKmerFreq[i].getKmer();
00260         kmer.normalize(validNucleotides);
00261         _vectorKmerFreq[i].setKmer(kmer);
00262     }
00263 }
00264
```

```
00265 void Profile::deletePos(int pos){
00266     if(pos<0 || pos>=_size){
00267         throw std::out_of_range(
00268         string("Profile::deletePos(int pos): ") +
00269                 "invalid position " + to_string(pos));
00270     }
00271     for(int i=pos+1; i<_size; i++){
00272         _vectorKmerFreq[i-1] = _vectorKmerFreq[i];
00273     }
00274     _size--;
00275 }
00276
00277 void Profile::zip(bool deleteMissing, int lowerBound) {
00278     int pos;
00279
00280     pos = 0;
00281     while (pos < _size) {
00282         if ((deleteMissing &&
00283             _vectorKmerFreq[pos].getKmer().toString().find(
00284                 Kmer::MISSING_NUCLEOTIDE) != string::npos) ||
00285             _vectorKmerFreq[pos].getFrequency() <= lowerBound) {
00286
00287             deletePos(pos);
00288         } else {
00289             pos++;
00290         }
00291     }
00292 }
00293
00294 KmerFreq& Profile::operator[](int index) const {
00295     return _vectorKmerFreq[index];
00296 }
00297
00298 KmerFreq& Profile::operator[](int index) {
00299     return _vectorKmerFreq[index];
00300 }
00301
00302 Profile& Profile::operator+=(const KmerFreq& kmerFreq){
00303     this->append(kmerFreq);
00304
00305     return *this;
00306 }
00307
00308 Profile& Profile::operator+=(const Profile& profile){
00309     for (int i = 0; i < profile.getSize(); i++) {
00310         (*this)+=profile.at(i);
00311     }
00312
00313     return *this;
00314 }
00315
00316 ostream& operator«(ostream& os, const Profile & language) {
00317     os « language.getProfileId() « endl;
00318     os « language.getSize() « endl;
00319     for (int i = 0; i < language.getSize(); ++i) {
00320         os «  language.at(i) « endl; //or os «  idioma[i] « endl;
00321     }
00322     return os;
00323 }
00324
00325 istream& operator»(istream& is, Profile& profile) {
00326     string speciesId;
00327     int numberKmers;
00328     KmerFreq kmerFreq;
00329
00330 //    profile.deallocate();
00331
00332     is » speciesId; // Read species identifier
00333     is » numberKmers; // Read the number of kmers
00334     if (numberKmers < 0) {
00335         throw std::out_of_range(
00336                 string("istream& operator»(istream& is, Profile& profile): ") +
00337                 "the number of kmers read " + to_string(numberKmers) + " is not valid ");
00338     }
00339 //    profile.allocate(numberKmers);
00340
00341     Profile iaux(numberKmers);
00342     iaux.setProfileId(speciesId);
00343 //    for (int i = 0; i < iaux.getSize(); ++i) {
00344     for (int i = 0; i < numberKmers; ++i) {
00345         is » kmerFreq;
00346 //        iaux.at(i) = kmerFreq; // or iaux[i] = bgr;
00347         iaux.append(kmerFreq);
00348     }
00349     profile = iaux;
00350     return is;
00351 }
```

```
00352
00353
00354 // Private methods
00355
00356 void Profile::allocate(int capacity) {
00357     if (capacity > 0) {
00358         _capacity = capacity;
00359     }
00360     else{
00361         _capacity = 0;
00362     }
00363
00364     if(_capacity>0){
00365         _vectorKmerFreq = new KmerFreq[_capacity]; // each pair contains "_", 0
00366     }
00367     else{
00368         _vectorKmerFreq = nullptr;
00369     }
00370     _size = 0;
00371 }
00372
00373 void Profile::reallocate(int newCapacity) {
00374     if (newCapacity != this->_capacity) {
00375         KmerFreq *newVector;
00376         if (newCapacity > 0) {
00377             _capacity = newCapacity;
00378         } else {
00379             _capacity = 0;
00380         }
00381         if(_capacity>0){
00382             newVector = new KmerFreq[_capacity];
00383         }
00384         else{
00385             newVector = nullptr;
00386         }
00387
00388         for (int i = 0; i < _capacity && i < _size; i++) {
00389             newVector[i] = _vectorKmerFreq[i];
00390         }
00391         delete[] _vectorKmerFreq;
00392         _vectorKmerFreq = newVector;
00393     }
00394 }
00395
00396 void Profile::deallocate() {
00397     delete[] _vectorKmerFreq;
00398     _vectorKmerFreq = nullptr;
00399     _size = 0;
00400     _capacity = 0;
00401 }
00402
00403 void Profile::copy(const Profile& otherProfile) {
00404     this->_profileId = otherProfile.getProfileId();
00405     this->_size = otherProfile._size; // En principio, no sería necesario si se cumple la precondición
00406     for (int i = 0; i < otherProfile.getSize(); ++i) {
00407         _vectorKmerFreq[i] = otherProfile.at(i); // or _vectorKmerFreq[i] = other[i];
00408     }
00409 }
```

Index