Gk-arrays

Generated by Doxygen 1.7.6.1

Thu Mar 21 2013 15:36:51

Contents

1 Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

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

3 Class Documentation

3.1 gkarrays::gkArrays Class Reference

Public Member Functions

- gkArrays (char *tags_file, uint threshold, bool use_bitvector=false, uint tag_length=0, bool stranded=false, uint nb_threads=1)
- gkArrays (char *tags_file1, char *tags_file2, uint threshold, bool use_bitvector=false, uint tag_length=0, bool stranded=false, uint nb_threads=1)
- uintSA convertPposToQpos (uintSA i)
- uintSA getEndPosOfTagNum (uint tag_num)
- uintSA getGkCFA (uintSA i)
- uintSA getGkCFALength ()
- uintSA getGkISA (uintSA i)
- uintSA getGkSA (uintSA i)
- uintSA getGkSALength ()
- uintSA getNbPposition (uintSA nb_reads)
- uint getNbTags ()

- uint getNbTagsWithFactor (uint tag_num, uint pos_factor, bool multiplicity=0)
- uint getNbThreads ()
- uint getPair (uint i)
- uintSA getPosInCommon (uint tag_num, uint pos_factor)
- readsReader * getReads ()
- uintSA getStartPosOfTagNum (uint tag_num)
- uintSA getStartQPosOfTagNum (uint tag_num)
- uint * getSupport (uint i)
- uint getSupportLength (uint i=0)
- char * getTag (uint i)
- uint getTagLength (uint i=0)
- char * getTagFactor (uint i, uint p, uint l)
- uint getTagNum (uintSA pos)
- std::pair< uint, uint > getTagNumAndPosFromAbsolutePos (uintSA pos)
- uint * getTagNumWithFactor (uint tag_num, uint pos_factor)
- std::pair< uint, uint > * getTagsWithFactor (uint tag_num, uint pos_factor)
- std::pair< uint, uint > * getTagsWithFactor (char *factor, uint factor_length, uint &nb_fact)
- char * getTextFactor (uintSA pos, uint length)
- uint getThreshold ()
- array_type getType ()
- bool isLarge ()
- bool isPposition (uintSA pos)
- bool isStranded ()
- bool isTheFirstMemberOfPair (uint i)

Static Public Member Functions

- static bool isDiscarded (uint actual_length, uint theoretical_length=0, uint k=0)

3.1.1 Constructor & Destructor Documentation

3.1.1.1 gkarrays::gkArrays::gkArrays (char * *tags_file*, uint *threshold*, bool *use_bitvector* = `false`, uint *tag_length* = `0`, bool *stranded* = `false`, uint *nb_threads* = `1`)

Construct the read index

Parameters

tags_file	Name of the file containg the reads
threshold	length of k-mers we have to use
use_bitvector,:	true iff we must store the array using a bit vector (slower but more space efficient)
tag_length	length of the reads. If a shorter read is found, it raises an error. If a longer read is found, only the prefix of tag_length characters is kept. If tag_length == 0 (default), just gess what the read length is.

stranded,:	true iff we know which strand has been sequenced and, therefore, (for instance) AACG must not be considered as equal to its revcomp (CGTT).
nb_threads	allows to build GkSA on a multi-thread architecture

3.1.1.2 gkarrays::gkArrays::gkArrays (char * *tags_file1,* char * *tags_file2,* uint *threshold,* bool *use_bitvector =* `false`, uint *tag_length =* `0`, bool *stranded =* `false`, uint *nb_threads =* `1`)

Alternative to construct the read index with paired-end reads

Parameters

tags_file1	Name of the file containing the reads of the first pair
tags_file2	Name of the file containing the reads of the second pair
threshold	length of k-mers we have to use
use_bitvector,:	true iff we must store the array using a bit vector (slower but more space efficient)
tag_length	length of the reads. If a shorter read is found, it raises an error. If a longer read is found, only the prefix of tag_length characters is kept. If tag_length == 0 (default), just gess what the read length is.
stranded,:	true iff we know which strand has been sequenced and, therefore, (for instance) AACG must not be considered as equal to its revcomp (CGTT).
nb_threads	allows to build GkSA on a multi-thread architecture

3.1.2 Member Function Documentation

3.1.2.1 uintSA gkarrays::gkArrays::convertPposToQpos (uintSA *i*)

Convert a position from P-position to Q-position (if you do not understand this, please read our article!). That converts a position as in the concatenation of reads to the position in GkIFA (for example). In the article, values of GkSA are also renumbered to Q-position but we do not renumber them in practice (it is quite useless).

Parameters

i,:	a P-position

Returns

a Q-position

3.1.2.2 uintSA gkarrays::gkArrays::getEndPosOfTagNum (uint *tag_num*)

Gives the end position of a given read in the concatenation of reads.

Parameters

tag_num,:	tag number

Returns

the end position of the read #tag_num in C_R (the concatenation of reads)

3.1.2.3 uintSA gkarrays::gkArrays::getGkCFA (uintSA *i*)

Parameters

i	the index position in the array (starting at 0).

Returns

the value of GkCFA at the given index ie. the number of k-factors of rank i, where i is the requested index.

3.1.2.4 uintSA gkarrays::gkArrays::getGkCFALength ( )

Returns

the number of elements in the GkCFA array. In other terms it corresponds to the number of distinct k-mers in the input.

3.1.2.5 uintSA gkarrays::gkArrays::getGkISA (uintSA *i*)

Parameters

i	the index position in the array (starting at 0).

Returns

the value of GkISA at the given index ie. the rank of the k-factor at position P-position i.

3.1.2.6 uintSA gkarrays::gkArrays::getGkSA (uintSA *i*)

Parameters

i	the index position in the array (starting at 0).

Returns

the value of GkSA at the given index ie. the P-position of the k-factor whose rank is i

3.1.2.7 uintSA gkarrays::gkArrays::getGkSALength ( )

Returns

the number of entries in gkSA (ie. the number of P-positions)

3.1.2.8 uintSA gkarrays::gkArrays::getNbPposition (uintSA *nb_reads*)

Returns

the number of P-positions in Cr from a number of reads (fixed length or not) This function is available before the construction of gkSA.

3.1.2.9 uint gkarrays::gkArrays::getNbTags ( )

Returns

the number of tags (or reads) indexed in the Gk Arrays

3.1.2.10 uint gkarrays::gkArrays::getNbTagsWithFactor (uint *tag_num*, uint *pos_factor*, bool *multiplicity* = 0)

Parameters

tag_num	The number of the tag in the index
pos_factor	Position of the factor in the tag
multiplicity	Counts (if false) only once a tag that contains the factor many times

Returns

Return the number of tags sharing the factor starting at position pos_factor in the tag tag_num. This is the number of elements returned by the function getTagsWithFactor(.)

3.1.2.11 uint gkarrays::gkArrays::getNbThreads ( )

Returns

the number of threads the GkArrays have been told to use. The threads can be used for the construction.

3.1.2.12 uint gkarrays::gkArrays::getPair (uint *i*)

Parameters

i	The number of the tag in the index

Returns

the tag number of the paired-end read associated with i or -1 if reads are not paired-end.

3.1.2.13 uintSA gkarrays::gkArrays::getPosInCommon (uint *tag_num,* uint *pos_factor*)

Returns

the rank of the P-k factor starting at position pos_factor in the read number tag_-num.

3.1.2.14 readsReader * gkarrays::gkArrays::getReads ( )

Returns

the object that allows to get a readIterator

3.1.2.15 uintSA gkarrays::gkArrays::getStartPosOfTagNum (uint *tag_num*)

Gives the start position of a given read in the concatenation of reads.

Parameters

tag_num,:	tag number

Returns

the start position of the read #tag_num in C_R (the concatenation of reads)

3.1.2.16 uintSA gkarrays::gkArrays::getStartQPosOfTagNum (uint *tag_num*)

Gives the start Q-position of a given read in the ISA array

Parameters

tag_num,:	tag number

Returns

the start Q-position of the read #tag_num in GkISA.

3.1.2.17 uint * gkarrays::gkArrays::getSupport (uint *i*)

Parameters

i	Tag number

Returns

an array whose length is getSupportLength(i) and where the value at position k is the number of occurrences of the k-factor starting at position k in the reads among all the Pk-factors.

3.1.2.18 uint gkarrays::gkArrays::getSupportLength (uint *i* = 0)

Return the length of the support.

Returns

getTagLength(i) - getThreshold()+1

3.1.2.19 char * gkarrays::gkArrays::getTag (uint *i*)

Parameters

i	the read number to be retrieved

Returns

the read number i.

3.1.2.20 char * gkarrays::gkArrays::getTagFactor (uint *i*, uint *p*, uint *l*)

Parameters

i	The number of the tag in the index
p	Position of the factor in the tag
l	The length of the factor

Returns

the factor at the position p in the tag number i

3.1.2.21 uint gkarrays::gkArrays::getTagLength (uint *i* = 0)

Parameters

i	Tag number (if the length is not constant)

Returns

the length of the read.

3.1.2.22 uint gkarrays::gkArrays::getTagNum (uintSA *pos*)

Gives the number of a read

Parameters

pos	a position in SA or in the concatenated sequence of reads

Returns

the read number where this position lies

3.1.2.23 std::pair< uint, uint > gkarrays::gkArrays::getTagNumAndPosFromAbsolutePos (uintSA *pos*)

Return the number of tag and the relative position in that tag corresponding to a given position in the concatenation of reads

Parameters

pos	position in the concatenation of reads

Returns

a pair whose fist element is the tag number and the second element is the position in the tag.

3.1.2.24 uint * gkarrays::gkArrays::getTagNumWithFactor (uint *tag_num,* uint *pos_factor*)

Parameters

tag_num	The number of the tag in the index
pos_factor	Position of the factor in the tag

Returns

Return an array that contains each tag number where the factors matches.

Postcondition

The array is sorted

3.1.2.25 pair< uint, uint > * gkarrays::gkArrays::getTagsWithFactor (uint *tag_num,* uint *pos_factor*)

Parameters

tag_num	The number of the tag in the index
pos_factor	Position of the factor in the tag

Returns

Return an array composed of pairs (tag, pos) corresponding to all the Pk-factors equal to the Pk-factor starting at position pos_factor in the tag tag_num.

Postcondition

The array is sorted according to read number and read position

3.1.2.26 pair< uint, uint > * gkarrays::gkArrays::getTagsWithFactor (char * *factor*, uint *factor_length*, uint & *nb_fact*)

Parameters

factor	the pattern to be searched.
factor_length	the length of the factor, should be <= getThreshold()
nb_fact	nb_fact is used to give the number of occurrences in the array.

Returns

Return an array composed of pairs (tag, pos) corresponding to all the Pk-factors equal to the k-factor factor

3.1.2.27 char * gkarrays::gkArrays::getTextFactor (uintSA *pos*, uint *length*)

Parameters

pos	The position from where we want to retrieve a text subtring. The position must be given in the original text (not the filtered one).
length	the length of the substring to be retrieved.

Returns

text factor at position pos of length length. The returned string is NULL-terminated.

3.1.2.28 uint gkarrays::gkArrays::getThreshold ( )

Returns

return the length of the k-factors (ie. return k).

3.1.2.29 array_type gkarrays::gkArrays::getType ( )

Returns

the array type used for building GkSA and GkISA (either SMALL_ARRAY, LARGE_ARRAY or OPTIMAL_ARRAY).

3.1.2.30 bool gkarrays::gkArrays::isDiscarded (uint *actual_length*, uint *theoretical_length* = 0, uint *k* = 0) [static]

Returns

true iff the read is not suitable ie. if it is shorter than the specified length (if any) or shorter than the specified k-mer length.

3.1.2.31 bool gkarrays::gkArrays::isLarge ( )

Returns

true if the nbPposition $> 2^{\wedge}32$

3.1.2.32 bool gkarrays::gkArrays::isPposition (uintSA *pos*)

Returns

true iff the position does not lie in the threshold - 1 last characters of a read, ie. if it is a P-position.

3.1.2.33 bool gkarrays::gkArrays::isStranded ( )

Returns

true iff the GkArrays have been built as a strand-dependant index. Therefore a k-mer and its revcomp won't be considered as equal.

3.1.2.34 bool gkarrays::gkArrays::isTheFirstMemberOfPair (uint *i*)

Parameters

i	the number of the tag in the index

Returns

true if the tag is the first member of is pair in case of paired-end files. False either

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

- gkArrays.h
- gkArrays.cpp

3.2 gkarrays::GkArraysBuilder Class Reference

Public Member Functions

- GkArraysBuilder (unsigned char ∗cr, uintSA length, int k, gkArrays ∗gk, array_type type, uint nb_threads)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 gkarrays::GkArraysBuilder::GkArraysBuilder (unsigned char ∗ *cr,* uintSA *length,* int *k,* gkArrays ∗ *gk,* array_type *type,* uint *nb_threads*)

Build the tables of the GkArrays for the input data.

Parameters

cr,:	the concatenation of reads
length,:	length of the read concatenation
k,:	length of the k-mers
gk,:	gkArrays to be used for computing gkIFA
type,:	the type of array to build
nb_threads,:	the number of threads that can be used

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

- gkArraysBuilder.h
- gkArraysBuilder.cpp

3.3 gkarrays::GkCFPSBuilder Class Reference

Public Member Functions

- void build (SolArray *values, uintSA length, array_type t, uint nb_threads=1)
- SolArray * getGkCFPS ()

3.3.1 Member Function Documentation

3.3.1.1 void gkarrays::GkCFPSBuilder::build (SolArray * *values,* uintSA *length,* array_type *t,* uint *nb_threads* = `1`)

Build a GkCFPS

Parameters

values,:	a sorted array containing k-mer integer values
length,:	length of the array values (number of elements)
t,:	the type of array to build
nb_threads,:	the number of threads that can be used

3.3.1.2 SolArray * gkarrays::GkCFPSBuilder::getGkCFPS ( )

Returns

the last GkCFPS that has been built

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

- gkCFPSBuilder.h
- gkCFPSBuilder.cpp

3.4 gkarrays::GkIFABuilder Class Reference

Public Member Functions

- void build (gkArrays *gk, SolArray *gkSA, SolArray *gkCFPS, array_type t, uint nb_threads=1)
- SolArray * getGkIFA ()

3.4.1 Member Function Documentation

3.4.1.1 void gkarrays::GkIFABuilder::build (gkArrays * *gk,* SolArray * *gkSA,* SolArray * *gkCFPS,* array_type *t,* uint *nb_threads* = 1)

Build a GkIFA

Parameters

gk,:	gkArrays to be used for computing gkIFA
gkSA,:	gkSA to be used for computing gkIFA
gkCFPS,:	gkCFPS to be used for computing gkIFA
t,:	the type of array to build
nb_threads,:	the number of threads that can be used

3.4.1.2 SolArray * gkarrays::GkIFABuilder::getGkIFA ( )

Returns

the last GkIFA that has been built

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

- gkIFABuilder.h
- gkIFABuilder.cpp

3.5 gkarrays::GkSABuilder Class Reference

Public Member Functions

- void build (SolArray *values, unsigned char *cr, uintSA length, int k, gkArrays *gk, int threshold_sort, array_type type, uint nb_threads)
- SolArray * getGkSA ()

3.5.1 Member Function Documentation

3.5.1.1 void gkarrays::GkSABuilder::build (SolArray * *values,* unsigned char * *cr,* uintSA *length,* int *k,* gkArrays * *gk,* int *threshold_sort,* array_type *type,* uint *nb_threads*)

Build a GkSA

Parameters

values,:	array storing the integer values of the k-mers
cr,:	the concatenation of reads
length,:	length of the read concatenation
k,:	length of the k-mers
gk,:	gkArrays to be used for computing gkIFA
threshold_sort,:	threshold below which an insertion sort is performed rather than a quicksort
type,:	the type of SolArray to build
nb_threads,:	the number of threads that can be used

3.5.1.2 SolArray * gkarrays::GkSABuilder::getGkSA ( )

Returns

the last GkSA that has been built.

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

- gkSABuilder.h
- gkSABuilder.cpp

3.6 gkarrays::ifa_builder_thread Struct Reference

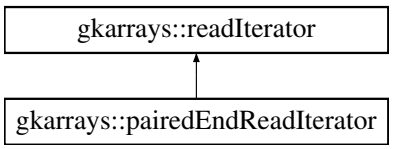
The documentation for this struct was generated from the following file:

- gkIFABuilder.h

3.7 gkarrays::pairedEndReadIterator Class Reference

`#include <readsReader.h>`

Inheritance diagram for gkarrays::pairedEndReadIterator:



Public Member Functions

- pairedEndReadIterator (char *filename1, char *filename2, uint k=0, uint length=0, bool printWarnings=false)
- pairedEndReadIterator (const pairedEndReadIterator &mit)
- virtual ~pairedEndReadIterator ()

- virtual readIterator & operator++ ()
- virtual pairedEndReadIterator operator++ (int)
- virtual kseq_t & operator* ()
- virtual size_t getLength ()
- virtual char * getName ()
- readIterator & getPair ()
- virtual char * getQuality ()
- virtual uint getReadNumber ()
- virtual char * getSequence ()
- virtual bool isFinished ()
- bool isTheFirstMemberOfPair ()

3.7.1 Detailed Description

pairedEndReadIterator allows to retrieve information on each read by traversing them in order, one after one in the case of paired-end reads. If the k-mer length or the read length are given, only reads that are long enough to fulfill those values are considered.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 gkarrays::pairedEndReadIterator::pairedEndReadIterator (char * *filename1*, char * *filename2*, uint *k* = `0`, uint *length* = `0`, bool *printWarnings* = `false`)

Constructor. Reads are retrieved from the given file.

Parameters

filename1,:	file we must iterate on
filename2,:	second file (for paired-end) we must iterate on
k,:	k-mer length used (0 if unknown or not applicable)
length,:	Read length (0 for variable)
printWarnings,:	if true print warnings when skipping a read

3.7.2.2 gkarrays::pairedEndReadIterator::pairedEndReadIterator (const pairedEndReadIterator & *mit*)

Copy constructor

3.7.2.3 gkarrays::pairedEndReadIterator::~pairedEndReadIterator ( ) `[virtual]`

Destructor

3.7.3 Member Function Documentation

3.7.3.1 size_t gkarrays::pairedEndReadIterator::getLength ( ) [virtual]

Returns

the length of the read

Implements gkarrays::readIterator.

3.7.3.2 char * gkarrays::pairedEndReadIterator::getName ( ) [virtual]

Returns

the name of the read

Implements gkarrays::readIterator.

3.7.3.3 readIterator & gkarrays::pairedEndReadIterator::getPair ( )

Returns

the paired iterator with the current one

3.7.3.4 char * gkarrays::pairedEndReadIterator::getQuality ( ) [virtual]

Returns

the quality of the read

Implements gkarrays::readIterator.

3.7.3.5 uint gkarrays::pairedEndReadIterator::getReadNumber ( ) [virtual]

Returns

the number of the read in the file

Implements gkarrays::readIterator.

3.7.3.6 char * gkarrays::pairedEndReadIterator::getSequence ( ) [virtual]

Returns

the sequence of the read

Implements gkarrays::readIterator.

3.7.3.7 bool gkarrays::pairedEndReadIterator::isFinished ( ) [virtual]

Returns

true iff we have read the whole file

Implements gkarrays::readIterator.

3.7.3.8 bool gkarrays::pairedEndReadIterator::isTheFirstMemberOfPair ( )

Returns

true if the current read is the first member of the pair

3.7.3.9 kseq_t & gkarrays::pairedEndReadIterator::operator* ( ) [virtual]

Returns

the kseq_t related to the current read

Implements gkarrays::readIterator.

3.7.3.10 readIterator & gkarrays::pairedEndReadIterator::operator++ ( ) [virtual]

Go to next sequence prefix

Implements gkarrays::readIterator.

3.7.3.11 pairedEndReadIterator gkarrays::pairedEndReadIterator::operator++ (int) [virtual]

Go to next sequence postfix

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

- readsReader.h
- readsReader.cpp

3.8 gkarrays::qs Struct Reference

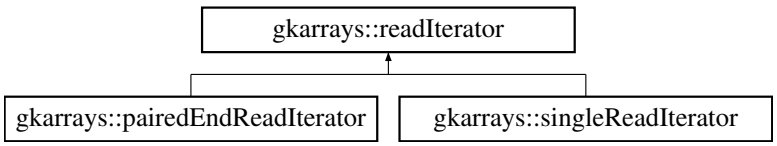
The documentation for this struct was generated from the following file:

- gkSABuilder.h

3.9 gkarrays::readIterator Class Reference

`#include <readsReader.h>`

Inheritance diagram for gkarrays::readIterator:



Public Member Functions

- virtual ~readIterator ()
- virtual readIterator & operator++ ()=0
- virtual kseq_t & operator* ()=0
- virtual size_t getLength ()=0
- virtual char * getName ()=0
- virtual char * getQuality ()=0
- virtual uint getReadNumber ()=0
- virtual char * getSequence ()=0
- virtual bool isFinished ()=0
- bool printWarnings ()
- void setPrintWarnings (bool isVisible)

3.9.1 Detailed Description

readIterator allows to retrieve information on each read by traversing them in order, one after one. If the k-mer length or the read length are given, only reads that are long enough to fulfill those values are considered.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 virtual gkarrays::readIterator::~readIterator ( ) [inline, virtual]

Virtual destructor, necessary in c++

3.9.3 Member Function Documentation

3.9.3.1 virtual size_t gkarrays::readIterator::getLength ( ) [pure virtual]

Returns

the length of the read

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.2 virtual char* gkarrays::readIterator::getName ( ) [pure virtual]

Returns

the name of the read

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.3 virtual char* gkarrays::readIterator::getQuality ( ) [pure virtual]

Returns

the quality of the read

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.4 virtual uint gkarrays::readIterator::getReadNumber ( ) [pure virtual]

Returns

the number of the read in the file

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.5 virtual char* gkarrays::readIterator::getSequence ( ) [pure virtual]

Returns

the sequence of the read

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.6 virtual bool gkarrays::readIterator::isFinished ( ) [pure virtual]

Returns

true iff we have read the whole file

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.7 virtual kseq_t& gkarrays::readIterator::operator* ( ) [pure virtual]

Returns

the kseq_t related to the current read

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.8 virtual readIterator& gkarrays::readIterator::operator++ ( ) [pure virtual]

Go to next sequence

Implemented in gkarrays::pairedEndReadIterator, and gkarrays::singleReadIterator.

3.9.3.9 bool gkarrays::readIterator::printWarnings ( )

Returns

true if warnings are visible false either.

3.9.3.10 void gkarrays::readIterator::setPrintWarnings (bool *isVisible*)

Parameters

isVisible	Printwarnins visible value.

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

- readsReader.h

- readsReader.cpp

3.10 gkarrays::readsReader Class Reference

`#include <readsReader.h>`

Public Member Functions

- readsReader (char *filename, uint k=0, uint length=0)
- readsReader (char *filename1, char *filename2, uint k=0, uint length=0)
- readIterator * begin (bool printWarnings=false)
- bool isPairedEnd ()

3.10.1 Detailed Description

readsReader is a class that allows you to store informations about an RNA_Seq experiment. It works both for single reads and paired-end reads. You can get an iterator using method begin() in order to go trough all reads.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 gkarrays::readsReader::readsReader (char * *filename*, uint *k* = 0, uint *length* = 0)

Single reads constructor

Parameters

filename	The name of the file containg reads
k	length of k-mers we have to use (0 for variable-length reads)
length	length of the reads. If a shorter read is found, it raises an error. If a longer read is found, only the prefix of tag_length characters is kept. If tag_length == 0 (default), just gess what the read length is.

3.10.2.2 gkarrays::readsReader::readsReader (char * *filename1*, char * *filename2*, uint *k* = 0, uint *length* = 0)

Single reads constructor

Parameters

filename1	The name of the first file containing reads
filename2	The name of the second file containing reads
k	length of k-mers we have to use (0 for variable-length reads)
length	length of the reads. If a shorter read is found, it raises an error. If a longer read is found, only the prefix of tag_length characters is kept. If tag_length == 0 (default), just gess what the read length is.

3.10.3 Member Function Documentation

3.10.3.1 readIterator * gkarrays::readsReader::begin (bool *printWarnings* = `false`)

Parameters

print-Warnings	Value of printWarnings option

Returns

an read iterator that goes through the read file(s)

3.10.3.2 bool gkarrays::readsReader::isPairedEnd ( )

Returns

true is the reads are paired-end (i.e there is two reads files)

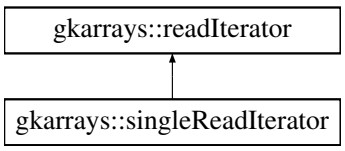
The documentation for this class was generated from the following files:

- readsReader.h
- readsReader.cpp

3.11 gkarrays::singleReadIterator Class Reference

`#include <readsReader.h>`

Inheritance diagram for gkarrays::singleReadIterator:



Public Member Functions

- singleReadIterator (char *filename, uint k=0, uint length=0, bool printWarnings=false, bool autoDiscard=true, bool autoFirstIteration=true)
- singleReadIterator (kseq_t *seq, uint k=0, uint length=0)
- singleReadIterator (const singleReadIterator &mit)
- ~singleReadIterator ()
- virtual readIterator & operator++ ()
- virtual singleReadIterator operator++ (int)
- virtual kseq_t & operator* ()
- virtual char * getName ()
- virtual char * getQuality ()

- virtual char * getSequence ()
- virtual uint getReadNumber ()
- virtual size_t getLength ()
- virtual bool isFinished ()
- bool isDiscarded ()
- void setAutoDiscard (bool value)

3.11.1 Detailed Description

singleReadIterator allows to retrieve information on each read by traversing them in order, one after one. If the k-mer length or the read length are given, only reads that are long enough to fulfill those values are considered.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 gkarrays::singleReadIterator::singleReadIterator (char * *filename,* uint *k* = 0, uint *length* = 0, bool *printWarnings* = false, bool *autoDiscard* = true, bool *autoFirstIteration* = true)

Constructor. Reads are retrieved from the given file.

Parameters

filename,:	file we must iterate on
k,:	k-mer length used (0 if unknown or not applicable)
length,:	Read length (0 for variable)
printWarnings,:	if true print warnings when skipping a read
autoDiscard,:	if true skip automatically bad reads
autoFirstIteration,:	automatically perform the first iteration (meaning that it initializes the data structure)

3.11.2.2 gkarrays::singleReadIterator::singleReadIterator (kseq_t * *seq,* uint *k* = 0, uint *length* = 0)

Constructor. singleReadIterator starts at the given sequence.

Parameters

seq,:	the sequence to start at.
k,:	k-mer length used (0 if unknown or not applicable)
length,:	Read length (0 for variable)

3.11.2.3 gkarrays::singleReadIterator::singleReadIterator (const singleReadIterator & *mit*)

Copy constructor

3.11.2.4 gkarrays::singleReadIterator::~singleReadIterator ( )

Destructor

3.11.3 Member Function Documentation

3.11.3.1 size_t gkarrays::singleReadIterator::getLength ( ) [virtual]

Returns

the length of the read

Implements gkarrays::readIterator.

3.11.3.2 char * gkarrays::singleReadIterator::getName ( ) [virtual]

Returns

the name of the read

Implements gkarrays::readIterator.

3.11.3.3 char * gkarrays::singleReadIterator::getQuality ( ) [virtual]

Returns

the quality of the read

Implements gkarrays::readIterator.

3.11.3.4 uint gkarrays::singleReadIterator::getReadNumber ( ) [virtual]

Returns

the number of the read in the file

Implements gkarrays::readIterator.

3.11.3.5 char * gkarrays::singleReadIterator::getSequence ( ) [virtual]

Returns

the sequence of the read

Implements gkarrays::readIterator.

3.11.3.6 bool gkarrays::singleReadIterator::isDiscarded ( )

Returns

true if real tag length is inferior to k or to wanted length

3.11.3.7 bool gkarrays::singleReadIterator::isFinished ( ) [virtual]

Returns

true iff we have read the whole file

Implements gkarrays::readIterator.

3.11.3.8 kseq_t & gkarrays::singleReadIterator::operator* ( ) [virtual]

Returns

the kseq_t related to the current read

Implements gkarrays::readIterator.

3.11.3.9 readIterator & gkarrays::singleReadIterator::operator++ ( ) [virtual]

Go to next sequence prefix

Implements gkarrays::readIterator.

3.11.3.10 singleReadIterator gkarrays::singleReadIterator::operator++ (int) [virtual]

Go to next sequence postfix

3.11.3.11 void gkarrays::singleReadIterator::setAutoDiscard (bool *value*)

Parameters

value	Value to set AutoDiscard mode

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

- readsReader.h
- readsReader.cpp

3.12 gkarrays::SolArray Class Reference

`#include <solArray.h>`

Public Member Functions

- SolArray ()
- SolArray (uintSA nbElements, array_type t)
- SolArray (void *ptr, array_type t, uintSA nbElements) throw (invalid_argument)

- SolArray (SolArray &sa)
- void init (uintSA nbElements, array_type t)
- uintSA get (uintSA pos)
- array_type getType ()
- uintSA length ()
- void realloc (uintSA nbElements) throw (logic_error)
- void set (uintSA pos, uintSA value)
- uintSA operator[] (uintSA pos)

3.12.1 Detailed Description

Small Or Large (SOL) Array storing integers. A SOL array can either store 32-bit, 64-bit integers, or fixed length integers on a different number of bits, depending on the user choice. This class allows to deal with the same variable whatever the type of integers we are storing.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 gkarrays::SolArray::SolArray ( )

Default constructor

Postcondition

getType() == SMALL_ARRAY && length() == 0

3.12.2.2 gkarrays::SolArray::SolArray (uintSA *nbElements*, array_type *t*)

Parameters

nbElements,:	number of elements to be stored in the array.
t,:	type of the array to be constructed. This specifies if we need to build a small array (32-bit integers) a large array (64-bit integers) or an optimal array which uses the optimal number of bits in memory (but which is longer)

3.12.2.3 gkarrays::SolArray::SolArray (void * *ptr*, array_type *t*, uintSA *nbElements*) throw (invalid_argument)

Parameters

ptr,:	pointer to an already allocated memory
t,:	type of the array to be built
nbElements,:	Number of elements that can be stored in the array

Precondition

t == SMALL_ARRAY || t == LARGE_ARRAY (doestn't work for bit vectors -- yet?)

3.12.2.4 gkarrays::SolArray::SolArray (SolArray & *sa*)

Copy constructor. Warning, the arrays are not copied, only the pointers are (to avoid huge memory consumption).

3.12.3 Member Function Documentation

3.12.3.1 uintSA gkarrays::SolArray::get (uintSA *pos*) [inline]

Parameters

pos,:	position in the array

Precondition

pos >= 0 && pos < length()

Returns

the value in the array at position pos

3.12.3.2 array_type gkarrays::SolArray::getType ( ) [inline]

Returns

the type of the SolArray that was constructed .

3.12.3.3 void gkarrays::SolArray::init (uintSA *nbElements,* array_type *t*)

Parameters

nb-Elements,:	number of elements to be stored in the array.
t,:	type of the array to be constructed

Precondition

The previous arrays must have been deleted if needed.

Postcondition

length() == nbElements && getType() == t

3.12.3.4 uintSA gkarrays::SolArray::length ( ) [inline]

Returns

The number of elements we can store in the array.

3.12.3.5 uintSA gkarrays::SolArray::operator[] (uintSA *pos*) [inline]

A shortcut to the get method.

3.12.3.6 void gkarrays::SolArray::realloc (uintSA *nbElements*) throw (logic_error)

Parameters

nbElements,:	number of elements the array must be reallocated to

Precondition

getType() != OPTIMAL_ARRAY

Postcondition

length() == nbElements AND the memory has been reallocated.

3.12.3.7 void gkarrays::SolArray::set (uintSA *pos,* uintSA *value*) [inline]

Parameters

pos,:	position in the array
value,:	value to be stored.

Precondition

pos >= 0 && pos < length()

Postcondition

get(pos) == value

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

- solArray.h
- solArray.cpp

3.13 gkarrays::subGkSA Struct Reference

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

- gkSABuilder.h