Package ‘stdvectors’

July 23, 2025

Type Package

Title C++ Standard Library Vectors in R

Version 0.0.5

Date 2017-02-20

Author Marco Giuliano

Maintainer Marco Giuliano <mgiuliano.mail@gmail.com>

Description Allows the creation and manipulation of C++ std::vector's in R.

License GPL (>= 2)

Imports Rcpp (>= 0.12.4)

URL https://github.com/digEmAll/stdvectors

BugReports https://github.com/digEmAll/stdvectors/issues

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-02-21 00:14:31

Contents

stdvectors-package *C++ Standard Library Vectors in R*

Description

Allows the creation and manipulation of C++ std::vector's in R.

Details

Package:	stdvectors
Type:	Package
Version:	0.0.5
Date:	2017-02-20
License:	GPL (>= 2)

This package allows the creation and manipulation of C++ std::vector's in R. std::vector's are dynamically allocated arrays, which are especially helpful when you need to fill a huge vector (e.g. in a loop) but you don't know the size in advance.

Author(s)

Marco Giuliano

Maintainer: Marco Giuliano <mgiuliano.mail@gmail.com>

References

cpp reference page : http://en.cppreference.com/w/

Examples

```
# create a stdvector
sv <- stdvectorCreate('integer')
# add 100 values to it
for(i in 1:100){
  # note that sv is modified in-place
  stdvectorPushBack(sv,i)
}
# get a normal R vector from the stdvector
v <- stdvectorToVector(sv)

## Not run:

  # check the time difference:
  # the first method takes around 2-3 s
  # the second method takes less than 0.1 s
  system.time({
```

```
    v <- integer()
    for(i in 1:100000){
      v[[length(v)+1]] <- i
    }
  }
)
system.time({
    v <- stdvectorCreate('integer')
    for(i in 1:100000){
      stdvectorPushBack(v,i)
    }
  }
)


## End(Not run)
```

---

stdvectorClass *std::vector R wrapper*

Description

Create and manipulate a C++ std:::vector in R.

Usage

```
stdvectorCreate(type = "double", reserve = 0L)
stdvectorPushBack(sdv, values)
stdvectorSize(sdv)
stdvectorClear(sdv)
stdvectorToVector(sdv)
stdvectorSubset(sdv,indexes)
stdvectorReplace(sdv,indexes,values)
stdvectorErase(sdv,indexFrom,indexTo)
stdvectorClone(sdv)
is.stdvector(x)
## S3 method for class 'stdvector'
print(x, ...)
## S3 method for class 'stdvector'
toString(x, ...)
```

Arguments

type	Character string indicating the type of the vector; possible values: double,numeric,integer,logical,c
reserve	The number of slots to be pre-allocated in the stdvector.
sdv	A stdvector object, as returned by stdvectorCreate.
...	optional arguments passed to inner print and toString methods. Unused.

x	A stdvector object, as returned by stdvectorCreate.
values	Values to be appended (in stdvectorPushBack) or set (in stdvectorReplace).
indexes	Indexes used to subset the current stdvector, in case of out of bounds indexes an error will be raised.
indexFrom	Used by stdvectorErase as starting index (inclusive) for the range of elements to be removed from stdvector.
indexTo	Used by stdvectorErase as ending index (inclusive) for the range of elements to be removed from stdvector.

Details

- stdvectorCreate creates a stdvector object of the indicated type.
- stdvectorPushBack appends elements to an existing stdvector (see note for type='any').
- stdvectorSize returns the number of elements of an existing stdvector.
- stdvectorClear removes all the elements of an existing stdvector.
- stdvectorToVector turns an existing stdvector into an R vector of the type chosen when the stdvector has been created.
- stdvectorSubset subsets an existing stdvector returning an R vector with the values corresponding to the selected indexes.
- stdvectorReplace replace the elements at indexes positions with the values in values argument (see note for type='any').
- stdvectorErase remove the elements from indexFrom to indexTo positions.
- stdvectorClone create a deep copy of the stdvector object.

Value

- stdvectorCreate returns an object of class stdvector.
- stdvectorPushBack return NULL invisibly.
- stdvectorSize returns an integer equal to the size of the stdvector.
- stdvectorClear returns NULL invisibly.
- stdvectorToVector returns an R vector of the type chosen when the stdvector has been created (type='any' will return a list).
- stdvectorSubset returns an R vector (of the type chosen when the stdvector has been created, type='any' will return a list) with the values corresponding to the selected indexes.
- stdvectorReplace returns NULL invisibly.
- stdvectorErase returns NULL invisibly.
- stdvectorClone returns an object of class stdvector which is the copy of the passed object.

Note

`stdvector`

- stdvector objects are treated as references, so if you do `sv2 <- sv1` and then you modify `sv2` actually also `sv1` will be modified. You need to do `sv2 <- stdvectorClone(sv1)` to actually create a copy.
- `stdvectorPushBack` in case of stdvector of `type='any'` will append the element passed in the argument `values` as a single new element of the vector, even if it's a list.
- `stdvectorSubset` indexes must be between 1 and the size of the stdvector.
- `stdvectorReplace` `indexes` and `values` must have the same length. In case of stdvector of `type='any'` will accept only `indexes` of length one.

References

See http://en.cppreference.com/w/cpp/container/vector

Examples

```
# create a stdvector
sv <- stdvectorCreate('integer')
# add 100 values to it
for(i in 1:100){
  # note that sv is modified in-place
  stdvectorPushBack(sv,i)
}
# get a normal R vector from the stdvector
v <- stdvectorToVector(sv)
```

Index