Simple vector class implementation on C++

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 BKVec< T > Class Template Reference

This class provides efficient, safe and convenient usage of arrays in C++.

`#include <BKVec.h>`

Public Member Functions

- BKVec ()
  Default constructor: BKVec()
- BKVec (size_t size)
  Constructor: BKVec(size_t)
- BKVec (const BKVec &v)
  Copy constructor: BKVec(const BKVec&)
- virtual ~BKVec ()
- size_t BKSize () const
- void BKPushBack (T const &v)
- T & operator[ ] (size_t idx) const
- BKVec & operator= (const BKVec &v)
- void BKClear ()

Private Member Functions

- void BKDeepCopy (const BKVec &v)
- void BKReserve ()
  Method for allocation memory BKReserve.

Private Attributes

- size_t fSize
  Size of the vector.
- size_t fCapacity
  Available capacity.
- T * fArray
  Pointer to the basic data types.

2.1.1 Detailed Description

template<class T>
class BKVec< T >

This class provides efficient, safe and convenient usage of arrays in C++.

Class: BKVec<T>

This class keeps an ordered list of values. It supports array selection ("[]"), but also supports inserting elements to the end.

Author

Borys Knysh

Version

Revision: 1.0

Date

2017/10/28

Contact: borys.knysh@gmail.com

2.1.2 Constructor & Destructor Documentation

2.1.2.1 BKVec() [1/3]

```
template<class T >
BKVec< T >::BKVec ( )
```

Default constructor: BKVec()

Usage: BKVec<T> bkVec Initializes a new vector. The default constructor creates an empty vector.

2.1.2.2 BKVec() [2/3]

```
template<class T >
BKVec< T >::BKVec (
            size_t size )
```

Constructor: BKVec(size_t)

Usage: BKVec<T> bkVec(size); Initializes a new vector. Creates array with given size of elements, which are initialized to zero.

2.1.2.3 BKVec() [3/3]

```
template<class T >
BKVec< T >::BKVec (
            const BKVec< T > & v )
```

Copy constructor: BKVec(const BKVec&)

Usage: BKVec<T> bkVec(bkVec2); Initializes a new vector. Creates the new vector from old one by assigning it size, capacity and values.

2.1.2.4 ~BKVec()

```
template<class T >
BKVec< T >::~BKVec ( )  [virtual]
```

Virtual destructor: ~BKVec() Frees any heap storage allocated by this vector.

2.1.3 Member Function Documentation

2.1.3.1 BKClear()

```
template<class T >
void BKVec< T >::BKClear ( )
```

Method: BKClear Usage: bkVec.BKClear(); This method is used for clearing vector.

2.1.3.2 BKDeepCopy()

```
template<class T >
void BKVec< T >::BKDeepCopy (
            const BKVec< T > & v )  [private]
```

Method: BKDeepCopy Usage: BKDeepCopy(bkVec); This private method is designed for replacing old vector on new vector.

Parameters

v	vector of basic data-type variables

2.1.3.3 BKPushBack()

```
template<class T >
```

```
void BKVec< T >::BKPushBack (
            T const & v )
```

Method: BKPushBack Usage: bkVec.BKPushBack(val);

Adds element to the end of the vector.

Parameters

v	value of basic data type

2.1.3.4 BKReserve()

```
template<class T >
void BKVec< T >::BKReserve ( )  [private]
```

Method for allocation memory BKReserve.

Usage: if(fSize == fCapacity) BKReserve(); This method allocates more memory for vector, if vector size reaches capacity level or vector is empty, while new elements are added, increases vector's capacity by chosing maximum value between 0 and double previous capacity.

2.1.3.5 BKSize()

```
template<class T>
size_t BKVec< T >::BKSize ( ) const  [inline]
```

Method: BKSize Usage: int nEl = (int)bkVec.BKSize();

Returns number of elements in this vector.

2.1.3.6 operator=()

```
template<class T >
BKVec< T > & BKVec< T >::operator= (
            const BKVec< T > & v )
```

Method: operator= Usage: bkVec2 = bkVec1; This method is used for assigning one vector to another.

Parameters

v	vector of basic data-type variables

2.1.3.7 operator[]()

```
template<class T >
T & BKVec< T >::operator[] (
             size_t idx ) const
```

Metod: operator[] Usage: bkVec[i] = 1; Overloads operator [], which is used to access vector elements.

Parameters

idx	index of element position in the vector

2.1.4 Member Data Documentation

2.1.4.1 fArray

```
template<class T>
T* BKVec< T >::fArray  [private]
```

Pointer to the basic data types.

This member class variable corresponds for keeping data in the “vector", could be any basic data types.

2.1.4.2 fCapacity

```
template<class T>
size_t BKVec< T >::fCapacity  [private]
```

Available capacity.

This member class varible contains number of allocated parts of memory for the given vector.

2.1.4.3 fSize

```
template<class T>
size_t BKVec< T >::fSize  [private]
```

Size of the vector.

Instance variables

This member class variable contains number of effectivly used parts of memory by user. Note: size value SHOULD be less or equal, than capacity value.

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

- BKVec.h

Index