TITLE

AUTHOR
Version 1.00
CREATEDATE

Table of Contents

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

File Index

File List

Here is a list of all files with brief descriptions:

Class Documentation

DateType Class Reference

```
#include <DateType.h>
```

Public Member Functions

- **DateType** ()
  Default constructor.
- **DateType** (char *newDate)
  Initialization constructor.

Public Attributes

- char **date** [**STD_STR_LEN**]

Static Public Attributes

- static const int **STD_STR_LEN** = 25

Constructor & Destructor Documentation

DateType::DateType ()

Default constructor.

Constructs empty **DateType**

Parameters:

None	

Note:

None

DateType::DateType (char * *newDate*)

Initialization constructor.

Constructs **DateType** with data components

Parameters:

in	new data, in string form

Note:

None

Member Data Documentation

char DateType::date[STD_STR_LEN]

const int DateType::STD_STR_LEN = 25 `[static]`

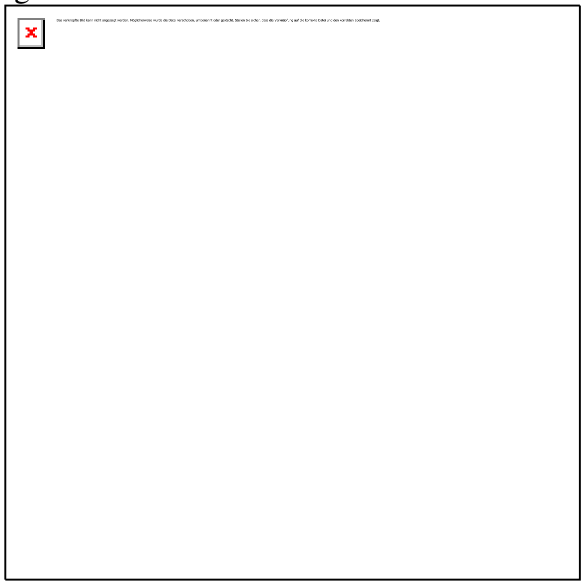
The documentation for this class was generated from the following files:

- **DateType.h**
- **DateType.cpp**

MrgSorter Class Reference

`#include <MrgSorter.h>`

Inheritance diagram for MrgSorter:



Public Member Functions

- **MrgSorter** ()
  Default constructor.
- **MrgSorter** (int initialCapacity)
  Initialization constructor.
- **MrgSorter** (const **SorterClass< DateType** > &copiedSorter)
  Copy constructor.
- virtual **~MrgSorter** ()
  Class destructor.
- virtual int **compareTo** (const **DateType** &lhObject, const **DateType** &rhObject)
  Object comparison, necessary for sorting.
- virtual bool **sort** ()
  Sorting operation.

Static Public Attributes

- static const char **NULL_CHAR** = '\0'
- static const char **SPACE** = ' '
- static const int **MONTH_NAME_WIDTH** = 3
- static const int **MAX_YEAR_ALLOWED** = 3000

Constructor & Destructor Documentation

MrgSorter::MrgSorter ()

Default constructor.

Constructs sorter class with default vector class initialization

Parameters:

None	

Note:

None

MrgSorter::MrgSorter (int *initialCapacity*)

Initialization constructor.

Constructs sorter class with specified vector class initialization

Parameters:

in	initial capacity

Note:

None

MrgSorter::MrgSorter (const SorterClass< DateType > & *copiedSorter*)

Copy constructor.

Constructs sorter class with copied object

Parameters:

in	other **SorterClass** object

Note:

None

MrgSorter::~MrgSorter () `[virtual]`

Class destructor.

Destructs test sorter class

Parameters:

in	None

Note:

Implements **SorterClass** -> **SimpleVector** destructor

Member Function Documentation

int MrgSorter::compareTo (const DateType & *lhObject*, const DateType & *rhObject*) `[virtual]`

Object comparison, necessary for sorting.

Compares objects mathematically, returns value < 0 if lhO < rhO returns 0 if lhO = rhO returns value > 0 if lhO > rhO

Parameters:

in	Left hand object, right hand object

Note:

Simple mathematical base operation; assumed to be overridden

Reimplemented from **SorterClass< DateType >** (*p.pagenum*).

bool MrgSorter::sort () `[virtual]`

Sorting operation.

Virtual sort method that is overridden to use various sorting strategies

Parameters:

in	None

Note:

Derived methods use specific strategy to sort objects

Sets sort success flag to true at start; supporting operations used to create dates, months, years will set the flag to false if there is an incorrect date; method returns success flag

Reimplemented from **SorterClass< DateType >** (*p.pagenum*).

Member Data Documentation

const int MrgSorter::MAX_YEAR_ALLOWED = 3000 `[static]`

const int MrgSorter::MONTH_NAME_WIDTH = 3 `[static]`

const char MrgSorter::NULL_CHAR = '\0' `[static]`

const char MrgSorter::SPACE = ' ' `[static]`

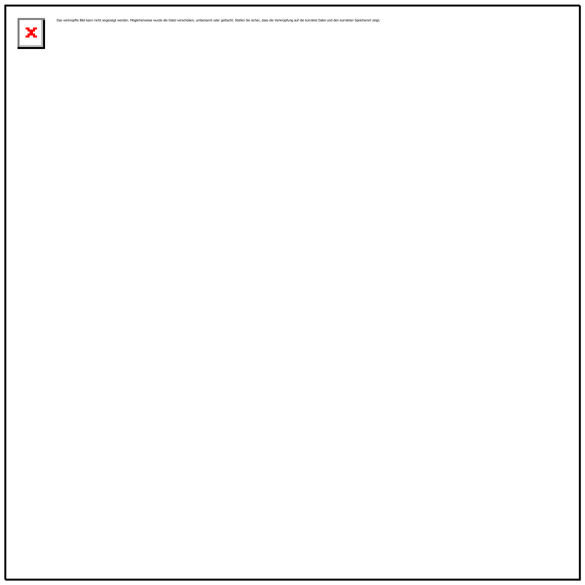
The documentation for this class was generated from the following files:

- **MrgSorter.h**
- **MrgSorter.cpp**

QkSorter Class Reference

`#include <QkSorter.h>`

Inheritance diagram for QkSorter:



Public Member Functions

- **QkSorter** ()
  Default constructor.
- **QkSorter** (int initialCapacity)
  Initialization constructor.
- **QkSorter** (const **SorterClass**< **DateType** > &copiedSorter)
  Copy constructor.
- virtual **~QkSorter** ()
  Class destructor.
- virtual int **compareTo** (const **DateType** &lhObject, const **DateType** &rhObject)
  Object comparison, necessary for sorting.
- virtual bool **sort** ()
  Sorting operation.

Static Public Attributes

- static const char **NULL_CHAR** = '\0'
- static const char **SPACE** = ' '
- static const int **MONTH_NAME_WIDTH** = 3
- static const int **MAX_YEAR_ALLOWED** = 3000

Constructor & Destructor Documentation

QkSorter::QkSorter ()

Default constructor.

Constructs sorter class with default vector class initialization

Parameters:

None	

Note:

None

QkSorter::QkSorter (int *initialCapacity*)

Initialization constructor.

Constructs sorter class with specified vector class initialization

Parameters:

in	initial capacity

Note:

None

QkSorter::QkSorter (const SorterClass< DateType > & *copiedSorter*)

Copy constructor.

Constructs sorter class with copied object

Parameters:

in	other **SorterClass** object

Note:

None

QkSorter::~QkSorter () `[virtual]`

Class destructor.

Destructs test sorter class

Parameters:

in	None

Note:

Implements **SorterClass** -> **SimpleVector** destructor

Member Function Documentation

int QkSorter::compareTo (const DateType & *lhObject*, const DateType & *rhObject*) `[virtual]`

Object comparison, necessary for sorting.

Compares objects mathematically, returns value < 0 if lhO < rhO returns 0 if lhO = rhO returns value > 0 if lhO > rhO

Parameters:

in	Left hand object, right hand object

Note:

Simple mathematical base operation; assumed to be overridden

Reimplemented from **SorterClass< DateType >** (*p.pagenum*).

bool QkSorter::sort () `[virtual]`

Sorting operation.

Virtual sort method that is overridden to use various sorting strategies

Parameters:

in	None

Note:

Derived methods use specific strategy to sort objects

Sets sort success flag to true at start; supporting operations used to create dates, months, years will set the flag to false if there is an incorrect date; method returns success flag

Reimplemented from **SorterClass< DateType >** (*p.pagenum*).

Member Data Documentation

const int QkSorter::MAX_YEAR_ALLOWED = 3000 `[static]`

const int QkSorter::MONTH_NAME_WIDTH = 3 `[static]`

const char QkSorter::NULL_CHAR = '\0' `[static]`

const char QkSorter::SPACE = ' ' `[static]`

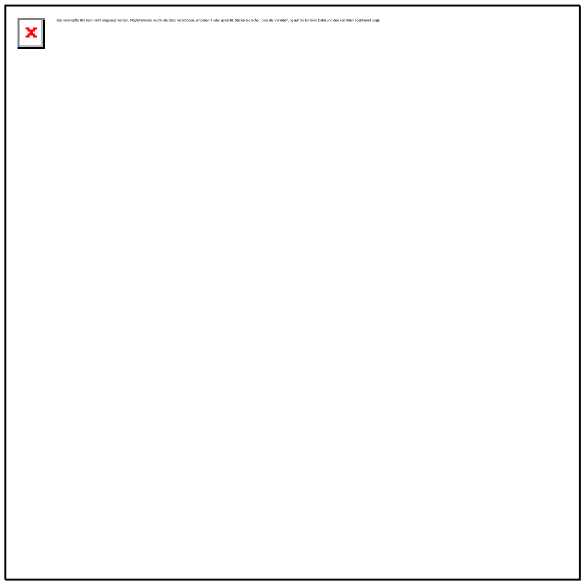
The documentation for this class was generated from the following files:

- **QkSorter.h**
- **QkSorter.cpp**

SelSorter Class Reference

`#include <SelSorter.h>`

Inheritance diagram for SelSorter:



Public Member Functions

- **SelSorter** ()
  Default constructor.
- **SelSorter** (int initialCapacity)
  Initialization constructor.
- **SelSorter** (const **SorterClass< DateType >** &copiedSorter)
  Copy constructor.
- virtual **~SelSorter** ()
  Class destructor.
- virtual int **compareTo** (const **DateType** &lhObject, const **DateType** &rhObject)
  Object comparison, necessary for sorting.
- virtual bool **sort** ()
  Sorting operation.

Static Public Attributes

- static const char **NULL_CHAR** = '\0'
- static const char **SPACE** = ' '
- static const int **MONTH_NAME_WIDTH** = 3
- static const int **MAX_YEAR_ALLOWED** = 3000

Constructor & Destructor Documentation

SelSorter::SelSorter ()

Default constructor.

Constructs sorter class with default vector class initialization

Parameters:

None	

Note:

None

SelSorter::SelSorter (int *initialCapacity*)

Initialization constructor.

Constructs sorter class with specified vector class initialization

Parameters:

in	initial capacity

Note:

None

SelSorter::SelSorter (const SorterClass< DateType > & *copiedSorter*)

Copy constructor.

Constructs sorter class with copied object

Parameters:

in	other **SorterClass** object

Note:

None

SelSorter::~SelSorter () `[virtual]`

Class destructor.

Destructs test sorter class

Parameters:

in	None

Note:

Implements **SorterClass** -> **SimpleVector** destructor

Member Function Documentation

int SelSorter::compareTo (const DateType & *lhObject*, const DateType & *rhObject*) `[virtual]`

Object comparison, necessary for sorting.

Compares objects mathematically, returns value < 0 if lhO < rhO returns 0 if lhO = rhO returns value > 0 if lhO > rhO

Parameters:

in	Left hand object, right hand object

Note:

Simple mathematical base operation; assumed to be overridden

Reimplemented from **SorterClass< DateType >** (*p.pagenum*).

bool SelSorter::sort () `[virtual]`

Sorting operation.

Virtual sort method that is overridden to use various sorting strategies

Parameters:

in	None

Note:

Derived methods use specific strategy to sort objects

Sets sort success flag to true at start; supporting operations used to create dates, months, years will set the flag to false if there is an incorrect date; method returns success flag

Reimplemented from **SorterClass< DateType >** (*p.pagenum*).

Member Data Documentation

const int SelSorter::MAX_YEAR_ALLOWED = 3000 `[static]`

const int SelSorter::MONTH_NAME_WIDTH = 3 `[static]`

const char SelSorter::NULL_CHAR = '\0' `[static]`

const char SelSorter::SPACE = ' ' `[static]`

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

- **SelSorter.h**
- **SelSorter.cpp**

SimpleTimer Class Reference

```
#include <SimpleTimer.h>
```

Public Member Functions

- **SimpleTimer** ()
  Default constructor.
- **~SimpleTimer** ()
  Default constructor.
- void **start** ()
  Start control.
- void **stop** ()
  Stop control.
- void **getElapsedTime** (char *timeStr)

Static Public Attributes

- static const char **NULL_CHAR** = '\0'
- static const char **RADIX_POINT** = '.'

Constructor & Destructor Documentation

SimpleTimer::SimpleTimer ()

Default constructor.

Constructs Timer class

Parameters:

None	

Note:

set running flag to false

SimpleTimer::~SimpleTimer ()

Default constructor.

Destructs Timer class

Parameters:

None	

Note:

No data to clear

Member Function Documentation

void SimpleTimer::getElapsedTime (char * *timeStr*)

void SimpleTimer::start ()

Start control.

Takes initial time data

Parameters:

None	

Note:

None

void SimpleTimer::stop ()

Stop control.

Takes final time data, calculates duration

Parameters:

None	

Note:

None

Member Data Documentation

const char SimpleTimer::NULL_CHAR = '\0' `[static]`

const char SimpleTimer::RADIX_POINT = '.' `[static]`

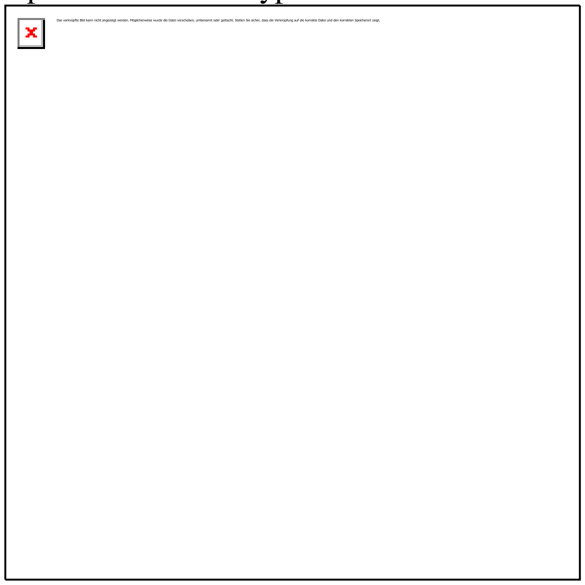
The documentation for this class was generated from the following files:

- **SimpleTimer.h**
- **SimpleTimer.cpp**

SimpleVector< DataType > Class Template Reference

`#include <SimpleVector.h>`

Inheritance diagram for SimpleVector< DataType >:



Public Member Functions

- **SimpleVector** ()
  Default constructor.
- **SimpleVector** (int newCapacity)
  Initialization constructor.
- **SimpleVector** (int newCapacity, const DataType &fillValue)
  Initialization constructor.
- **SimpleVector** (const **SimpleVector** &copiedVector)
  Copy constructor.
- **~SimpleVector** ()
  object destructor
- const **SimpleVector** & **operator=** (const **SimpleVector** &rhVector)
  assignment operation overload
- int **getCapacity** () const
  vector capacity accessor
- int **getSize** () const
  vector size accessor
- DataType & **operator[]** (int index) throw (logic_error)
  vector overloaded bracket operation
- const DataType & **operator[]** (int index) const throw (logic_error)
  vector overloaded bracket operation
- void **setValueAt** (int index, const DataType &item) throw (logic_error)
  vector data setting operation
- void **getValueAt** (int index, DataType &item) const throw (logic_error)
  vector data getting operation

- void **grow** (int growBy)
  vector resize larger operation
- void **shrink** (int shrinkBy) throw (logic_error)
  vector resize smaller operation
- void **incrementSize** ()
  vector size mutator - increase
- void **decrementSize** ()
  vector size mutator - decrease

Static Public Attributes

- static const int **DEFAULT_CAPACITY** = 10

Constructor & Destructor Documentation

template<class DataType > SimpleVector< DataType >::SimpleVector ()

Default constructor.

Constructs vector capacity to default and vector size to zero creates default size data array

Parameters:

None	

Note:

None

template<class DataType > SimpleVector< DataType >::SimpleVector (int *newCapacity*)

Initialization constructor.

Constructs vector capacity to given capacity and vector size to zero creates array of given capacity size

Parameters:

in	capacity with which to initialize vector

Note:

None

template<class DataType> SimpleVector< DataType >::SimpleVector (int *newCapacity*, const DataType & *fillValue*)

Initialization constructor.

Constructs vector to given capacity and zero size and sets each element to given fill value

Parameters:

in	capacity with which to initialize vector
in	fill value with which to initialize each element

Note:

None

template<class DataType> SimpleVector< DataType >::SimpleVector (const SimpleVector< DataType > & *copiedVector*)

Copy constructor.

Constructs vector capacity to default and vector size to zero creates default size data array

Parameters:

in	Other vector with which this vector is constructed

Note:

Uses copyVector to move data into this vector

template<class DataType > SimpleVector< DataType >::~SimpleVector ()

object destructor

If capacity is greater than zero, releases memory to system

Parameters:

None	

Note:

None

Member Function Documentation

template<class DataType > void SimpleVector< DataType >::decrementSize ()

vector size mutator - decrease

decreases vector size count

Parameters:

None	

Note:

has no effect on operation of vector; provided as convenience to user/programmer

template<class DataType > int SimpleVector< DataType >::getCapacity () const

vector capacity accessor

returns capacity of this vector

Parameters:

None	

Note:

None

template<class DataType > int SimpleVector< DataType >::getSize () const

vector size accessor

returns size of this vector

Parameters:

None	

Note:

None

template<class DataType> void SimpleVector< DataType >::getValueAt (int *index*, DataType & *item*) const throw logic_error)

vector data getting operation

allows direct access of the data from the vector

Parameters:

in	index of element to be assigned
in	data item to be retrieved from array

Note:

throws logic error if index is out of bounds

template<class DataType > void SimpleVector< DataType >::grow (int *growBy*)

vector resize larger operation

increases vector capacity by amount given in parameter

Parameters:

in	delta size for growth of vector

Note:

creates new data list, copies using copyVector, then deletes old list

template<class DataType > void SimpleVector< DataType >::incrementSize ()

vector size mutator - increase

increases vector size count

Parameters:

None	

Note:

has no effect on operation of vector; provided as convenience to user/programmer

template<class DataType > const SimpleVector< DataType > & SimpleVector< DataType >::operator= (const SimpleVector< DataType > & *rhVector*)

assignment operation overload

Assigns data from right-hand object to this object

Parameters:

in	right-hand vector object

Note:

Uses copyVector to move data into this vector

template<class DataType > DataType & SimpleVector< DataType >::operator[] (int *index*) throw logic_error)

vector overloaded bracket operation

allows assignment of data to element in this vector

Parameters:

in	index of element to be assigned

Note:

throws logic error if index is out of bounds

template<class DataType > const DataType & SimpleVector< DataType >::operator[] (int *index*) const throw logic_error)

vector overloaded bracket operation

allows assignment of data from element in this vector

Parameters:

in	index of element to be assigned

Note:

throws logic error if index is out of bounds

template<class DataType> void SimpleVector< DataType >::setValueAt (int *index*, const DataType & *item*) throw logic_error)

vector data setting operation

allows assignment of data directly to the vector

Parameters:

in	index of element to be assigned
in	data item to be stored in array

Note:

throws logic error if index is out of bounds

template<class DataType > void SimpleVector< DataType >::shrink (int *shrinkBy*) throw logic_error)

vector resize smaller operation

decreases vector capacity by amount given in parameter

Parameters:

in	delta size for reduction of vector

Note:

creates new data list, copies using copyVector, then deletes old list
vector does not check size before capacity reduction; if capacity is reduced to less than size, data will be lost

Member Data Documentation

template<class DataType> const int SimpleVector< DataType >::DEFAULT_CAPACITY = 10`[static]`

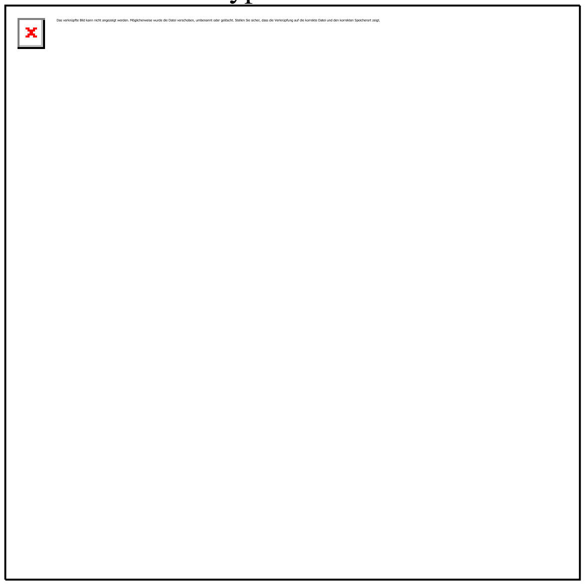
The documentation for this class was generated from the following files:

- **SimpleVector.h**
- **SimpleVector.cpp**

SorterClass< DataType > Class Template Reference

`#include <SorterClass.h>`

Inheritance diagram for SorterClass< DataType >:


Das verknüpfte Bild kann nicht angezeigt werden. Möglicherweise wurde die Datei verschoben, umbenannt oder gelöscht. Stellen Sie sicher, dass die Verknüpfung auf die korrekte Datei und den korrekten Speicherort zeigt.

Public Member Functions

- **SorterClass** ()
  Default constructor.
- **SorterClass** (int initialCapacity)
  Initialization constructor.
- **SorterClass** (const **SorterClass**< DataType > &copiedSorter)
  Copy constructor.
- virtual **~SorterClass** ()
  Class destructor.
- virtual void **add** (const DataType &addedObject)
  add item to sorter list
- virtual int **compareTo** (const DataType &lhObject, const DataType &rhObject)
  Object comparison, necessary for sorting.
- virtual bool **sort** ()
  Sorting operation.

Additional Inherited Members

Constructor & Destructor Documentation

template<typename DataType > SorterClass< DataType >::SorterClass ()

Default constructor.

Constructs sorter class with default vector class initialization

Parameters:

None	

Note:

None

template<typename DataType > SorterClass< DataType >::SorterClass (int *initialCapacity*)

Initialization constructor.

Constructs sorter class with specified vector class initialization

Parameters:

in	initial capacity

Note:

None

template<typename DataType> SorterClass< DataType >::SorterClass (const SorterClass< DataType > & *copiedSorter*)

Copy constructor.

Constructs sorter class with copied object

Parameters:

in	other **SorterClass** object

Note:

None

template<typename DataType > SorterClass< DataType >::~SorterClass () `[virtual]`

Class destructor.

Destructs sorter class

Parameters:

in	None

Note:

Implements **SimpleVector** destructor

Member Function Documentation

template<typename DataType> void SorterClass< DataType >::add (const DataType & *addedObject*) `[virtual]`

add item to sorter list

adds item to list for sorting

Parameters:

in	object to be added

Note:

None

template<typename DataType> int SorterClass< DataType >::compareTo (const DataType & *lhObject*, const DataType & *rhObject*)`[virtual]`

Object comparison, necessary for sorting.

Compares objects mathematically, returns value < 0 if lhO < rhO returns 0 if lhO = rhO returns value > 0 if lhO > rhO

Parameters:

in	Left hand object, right hand object

Note:

Simple mathematical base operation; assumed to be overridden

Reimplemented in **MrgSorter** (*p.pagenum*), **QkSorter** (*p.pagenum*), and **SelSorter** (*p.pagenum*).

template<typename DataType > bool SorterClass< DataType >::sort ()`[virtual]`

Sorting operation.

Virtual sort method that can be overridden to use various sorting strategies

Parameters:

in	None

Note:

None, virtual method takes no action, assumed to be overridden

Reimplemented in **MrgSorter** (*p.pagenum*), **QkSorter** (*p.pagenum*), and **SelSorter** (*p.pagenum*).

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

- **SorterClass.h**
- **SorterClass.cpp**

File Documentation

DateType.cpp File Reference

Implementation file for **DateType** class.

```
#include "DateType.h"
#include <cstring>
```

Macros

- #define **CLASS_DATETYPE_CPP**

Functions

- ostream & **operator<<** (ostream &outStream, const **DateType** &dateItem)
  ostream output operator

Detailed Description

Implementation file for **DateType** class.

Implements the constructor method of the **DateType** class

Author:
Michael Leverington

Version:
1.00 (11 September 2015)

Requires **DateType.h**

Macro Definition Documentation

#define CLASS_DATETYPE_CPP

Function Documentation

ostream& operator<< (ostream & *outStream*, const DateType & *dateItem*)

ostream output operator

Free function outputs **DateType** to stream

Parameters:

in	ostream file object
in	**DateType** data item

Note:
None

DateType.h File Reference

Definition file for **DateType** class.
`#include <ostream>`

Classes

- class **DateType**

Functions

- ostream & **operator<<** (ostream &outStream, const **DateType** &dateItem)
  ostream output operator

Detailed Description

Definition file for **DateType** class.

Specifies all data of the **DateType** class, along with the constructor, **DateType** class is entered and stored as a string

Author:
Michael Leverington

Version:
1.00 (11 September 2015)

None

Function Documentation

ostream& operator<< (ostream & *outStream*, const DateType & *dateItem*)

ostream output operator

Free function outputs **DateType** to stream

Parameters:

in	ostream file object
in	**DateType** data item

Note:
None

MrgSorter.cpp File Reference

```
#include "MrgSorter.h"
#include "SorterClass.cpp"
#include "SimpleVector.cpp"
```

MrgSorter.h File Reference

```
#include "DateType.h"
#include "SorterClass.h"
```

Classes

- class **MrgSorter**

PA05.cpp File Reference

```
#include "DateType.h"
#include "SimpleVector.cpp"
#include "SorterClass.cpp"
#include "SelSorter.h"
#include "MrgSorter.h"
#include "QkSorter.h"
#include "SimpleTimer.h"
#include <cstring>
#include <iostream>
```

Functions

- bool **getALine** (istream &consoleIn, char *str)
  Gets dates in three parts, combines to one string.
- void **displayList** (const **SelSorter** &dates, char dispID, bool sorted)
  Displays dates in order held.
- int **main** ()

Variables

- const int **SMALL_STR_LEN** = 25
- const int **DISPLAY_WIDTH_COUNT** = 5
- const int **SORTER_ITEMS** = 3
- const char **BREAK** [] = " - "
- const char **ENDLINE_CHAR** = '\n'
- const char **NULL_CHAR** = '\0'
- const char **COLON** = ':'
- const bool **SORTED** = true
- const bool **UNSORTED** = false

Function Documentation

void displayList (const SelSorter & *dates*, char *dispID*, bool *sorted*)

Displays dates in order held.

dates are displayed in a formatted way so they do not take as much vertical space

Parameters:

in	InsSorter object

Note:

virtual method uses specific strategy to sort objects

bool getALine (istream & *consoleIn*, char * *str*)

Gets dates in three parts, combines to one string.

dates are input using cin, and then recombined for string accommodates testing (Submit) system

Parameters:

in	istream object

out	string with date

Note:

resolution for redirected input, getline did not work

int main ()

Variable Documentation

const char BREAK[] = " - "

const char COLON = ':'

const int DISPLAY_WIDTH_COUNT = 5

const char ENDLINE_CHAR = '\n'

const char NULL_CHAR = '\0'

const int SMALL_STR_LEN = 25

const bool SORTED = true

const int SORTER_ITEMS = 3

const bool UNSORTED = false

PA05_New.cpp File Reference

```
#include "DateType.h"
#include "SimpleVector.cpp"
#include "SorterClass.cpp"
#include "SelSorter.h"
#include "MrgSorter.h"
#include "QkSorter.h"
#include "SimpleTimer.h"
#include <cstring>
#include <iostream>
```

Functions

- bool **getALine** (istream &consoleIn, char *str)
  Gets dates in three parts, combines to one string.
- void **displayList** (const InsSorter &dates, char dispID, bool sorted)
  Displays dates in order held.
- int **main** ()

Variables

- const int **SMALL_STR_LEN** = 25
- const int **DISPLAY_WIDTH_COUNT** = 5
- const int **SORTER_ITEMS** = 3
- const char **BREAK** [] = " - "
- const char **ENDLINE_CHAR** = '\n'
- const char **NULL_CHAR** = '\0'
- const char **COLON** = ':'
- const bool **SORTED** = true
- const bool **UNSORTED** = false

Function Documentation

void displayList (const InsSorter & *dates*, char *dispID*, bool *sorted*)

Displays dates in order held.

dates are displayed in a formatted way so they do not take as much vertical space

Parameters:

in	InsSorter object

Note:

virtual method uses specific strategy to sort objects

bool getALine (istream & *consoleIn*, char * *str*)

Gets dates in three parts, combines to one string.

dates are input using cin, and then recombined for string accommodates testing (Submit) system

Parameters:

in	istream object

out	string with date

Note:

resolution for redirected input, getline did not work

int main ()

Variable Documentation

const char BREAK[] = " - "

const char COLON = ':'

const int DISPLAY_WIDTH_COUNT = 5

const char ENDLINE_CHAR = '\n'

const char NULL_CHAR = '\0'

const int SMALL_STR_LEN = 25

const bool SORTED = true

const int SORTER_ITEMS = 3

const bool UNSORTED = false

QkSorter.cpp File Reference

```
#include "QkSorter.h"
#include "SorterClass.cpp"
#include "SimpleVector.cpp"
```

QkSorter.h File Reference

```
#include "DateType.h"
#include "SorterClass.h"
```

Classes

- class **QkSorter**

SelSorter.cpp File Reference

Implementation file for **SelSorter** using insertion sort, derived from **SorterClass**.

```
#include "SelSorter.h"
#include "SorterClass.cpp"
#include "SimpleVector.cpp"
```

Macros

- #define **SELSORTER_CPP**

Detailed Description

Implementation file for **SelSorter** using insertion sort, derived from **SorterClass**.

Author:

Mitchell Reyes

Implements virtual member methods of the **SelSorter**

Version:

1.00 (29 September 2015)

Requires **MrgSorter.h**, **SorterClass.cpp**, **SimpleVector.cpp**,

Author:

Mitchell Reyes

Implements virtual member methods of the **QkSorter**

Version:

1.00 (29 September 2015)

Requires **QkSorter.h**, **SorterClass.cpp**, **SimpleVector.cpp**,

Author:

Michael Leverington

Implements virtual member methods of the **SelSorter**

Version:

1.00 (11 September 2015)

Requires **SelSorter.h**, **SorterClass.cpp**, **SimpleVector.cpp**,

---

Macro Definition Documentation

#define SELSORTER_CPP

SelSorter.h File Reference

Definition file for **SelSorter** class using insertion sort, derived from **SorterClass**.

```
#include "DateType.h"
#include "SorterClass.h"
```

Classes

- class **SelSorter**

Detailed Description

Definition file for **SelSorter** class using insertion sort, derived from **SorterClass**.

Author:

Mitchell Reyes

Specifies all member methods of the **SelSorter** Class

Version:

1.00 (29 September 2015)

Requires **DateType.h**, **SorterClass.h**

Author:

Michael Leverington

Specifies all member methods of the **SelSorter** Class

Version:

1.00 (11 September 2015)

Requires **DateType.h**, **SorterClass.h**

SimpleTimer.cpp File Reference

Implementation file for **SimpleTimer** class.
`#include "SimpleTimer.h"`

Macros

- #define **SIMPLETIMER_CPP**

Detailed Description

Implementation file for **SimpleTimer** class.

Author:

Michael Leverington

Implements member methods for timing

Version:

1.00 (11 September 2015)

Requires **SimpleTimer.h**.

Macro Definition Documentation

#define SIMPLETIMER_CPP

SimpleTimer.h File Reference

Definition file for simple timer class.

```
#include <sys/time.h>
#include <cstring>
```

Classes

- class **SimpleTimer**

Detailed Description

Definition file for simple timer class.

Author:

Michael Leverington

Specifies all member methods of the **SimpleTimer**

Version:

1.00 (11 September 2015)

None

SimpleVector.cpp File Reference

Implementation file for **SimpleVector** class.
`#include "SimpleVector.h"`

Macros

- #define **CLASS_SIMPLEVECTOR_CPP**

Detailed Description

Implementation file for **SimpleVector** class.

Author:

Michael Leverington

Implements all member methods of the **SimpleVector** class

Version:

1.10 (11 September 2015) added getter and setter for date elements 1.00 (30 August 2015) origination

Requires **SimpleVector.h**

Macro Definition Documentation

#define CLASS_SIMPLEVECTOR_CPP

SimpleVector.h File Reference

Definition file for **SimpleVector** class.

```
#include <stdexcept>
```

Classes

- class **SimpleVector< DataType >**

Detailed Description

Definition file for **SimpleVector** class.

Author:

Michael Leverington

Specifies all member methods of the **SimpleVector** class

Version:

1.00 (11 September 2015)

None

SorterClass.cpp File Reference

Implementation file for **SorterClass**.

```
#include "SorterClass.h"
#include "SimpleVector.h"
```

Macros

- #define **SORTERCLASS_CPP**

Detailed Description

Implementation file for **SorterClass**.

Author:

Michael Leverington

Implements all member methods of the **SorterClass**

Version:

1.00 (11 September 2015)

Requires **SorterClass.h**, **SimpleVector.h**

Macro Definition Documentation

#define SORTERCLASS_CPP

SorterClass.h File Reference

Definition file for Sorter class.
`#include "SimpleVector.h"`

Classes

- class **SorterClass< DataType >**

Detailed Description

Definition file for Sorter class.

Author:

Michael Leverington

Specifies all member methods of the **SorterClass**

Version:

1.00 (11 September 2015)

Requires **SimpleVector.h**

Index

INDEX