Project 4

Tanner Jones
1.0 Version
9/23/2015

Table of Contents

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 documented files with brief descriptions:

Class Documentation

DateType Class Reference

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

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

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

SimpleTimer Class Reference

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::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:

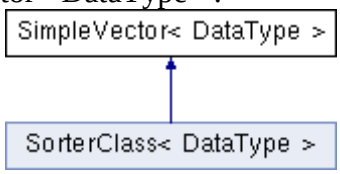
None

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

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

SimpleVector< DataType > Class Template Reference

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

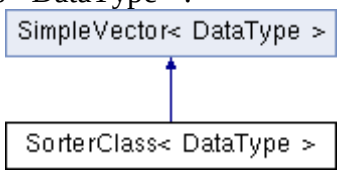
---

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

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

SorterClass< DataType > Class Template Reference

Inheritance diagram for SorterClass< DataType >:



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 **TestSorter** (*p.17*).

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

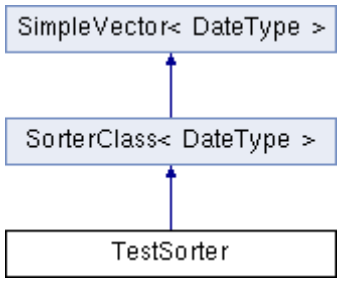
Reimplemented in **TestSorter** (*p.19*).

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

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

TestSorter Class Reference

Inheritance diagram for TestSorter:



Public Member Functions

- **TestSorter** ()
  Default constructor.
- **~TestSorter** ()
  ~TestSorter
- void **extractionLoop** (const **DateType** object, char *dummy)
  extractionLoop
- void **extraction** (int &day, char *month, int &year, const **DateType** &object)
  Default extraction.
- virtual int **compareTo** (const **DateType** &lhObject, const **DateType** &rhObject)
  compareTo
- int **findIndexofLargest** (int size)
  findIndexofLargest
- virtual bool **sort** ()
  sort
- void **dateSwitcher** (int lhObject, int rhObject)
  dateSwitcher
- int **charToInt** (char *number)
  charToInt
- void **lowerToUpper** (char *month)
  lowerToUpper
- bool **myCompare** (const char *stringOne, const char *stringTwo)
  myCompare
- int **monthNumberTester** (char *month)
  monthNumberTest

Additional Inherited Members

Constructor & Destructor Documentation

TestSorter::TestSorter ()

Default constructor.

Constructs of test sorter class with default vector class initialization

Parameters:

None	

Note:

None

TestSorter::~TestSorter ()

~TestSorter

deconstruct of test sorter class

Parameters:

None	

Note:

None

Member Function Documentation

int TestSorter::charToInt (char * *number*)

charToInt

switchs an char array to an int

Parameters:

char	array

Note:

None

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

compareTo

compares two different dataTypes and figures out which one is greater

Parameters:

simple	vector has already been filled

Note:

None

Reimplemented from **SorterClass< DateType >** (*p.14*).

void TestSorter::dateSwitcher (int *lhObject*, int *rhObject*)

dateSwitcher

takes two dates and switches them

Parameters:

two	different Dates

Note:

None

void TestSorter::extraction (int & *day*, char * *month*, int & *year*, const DateType & *object*)

Default extraction.

Takes apart the date and splits into three different test subjects

Parameters:

Simple	vector has already been filled

Note:

None

void TestSorter::extractionLoop (const DateType *object*, char * *dummy*)

extractionLoop

loops through date and copying the string

Parameters:

a	**DateType** and a char array

Note:

None

int TestSorter::findIndexofLargest (int *size*)

findIndexofLargest

goes through the whole vector and finds the largest value

Parameters:

simple	vector has already been filled

Note:

size

void TestSorter::lowerToUpper (char * *month*)

lowerToUpper

switchs a char from lower to upper case

Parameters:

an	array that holds the months

Note:

None

int TestSorter::monthNumberTester (char * *month*)

monthNumberTest

sets a value for each month

Parameters:

an	array for the month

Note:

None

bool TestSorter::myCompare (const char * *stringOne*, const char * *stringTwo*)

myCompare

compares two strings to see if they are similar

Parameters:

to	char arrays

Note:

None

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

sort

goes through the whole vector and sorts

Parameters:

simple	vector has already been filled

Note:

None

Reimplemented from **SorterClass< DateType >** (*p.15*).

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

- **TestSorter.h**
- **TestSorter.cpp**

File Documentation

DateType.cpp File Reference

Implementation file for **DateType** class.

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

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**

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

PA04.cpp File Reference

Driver program to exercise the DateSorter class.

```
#include "DateType.h"
#include "SimpleVector.cpp"
#include "SorterClass.cpp"
#include "TestSorter.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 **TestSorter** &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 char **BREAK** [] = " - "
- const char **ENDLINE_CHAR** = '\n'
- const char **NULL_CHAR** = '\0'
- const char **COLON** = ':'
- const bool **SORTED** = true
- const bool **UNSORTED** = false

Detailed Description

Driver program to exercise the DateSorter class.

Allows for testing the DateSorter class, along with a timer class that will be used for evaluation

Version:

1.00 (11 September 2015)

Requires **DateType.h**, **SimpleVector.cpp**, **SorterClass.cpp**, **TestSorter.h** cstring and iostream libraries

---

Function Documentation

void displayList (const TestSorter & *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	**TestSorter** 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

SimpleTimer.cpp File Reference

Implementation file for **SimpleTimer** class.

```
#include "SimpleTimer.h"
```

Detailed Description

Implementation file for **SimpleTimer** class.

Author:

Michael Leverington

Implements member methods for timing

Version:

1.00 (11 September 2015)

Requires **SimpleTimer.h**.

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"
```

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**

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"
```

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**

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**

TestSorter.cpp File Reference

Header file for the Test Sorter class.

```
#include "TestSorter.h"
#include "SorterClass.h"
```

Variables

- const char **WHITE_SPACE** = ' '
- const char **NULL_CHAR** = '\0'
- const char **THREE** = 3
- const char **JAN** [] = "JAN"
- const char **FEB** [] = "FEB"
- const char **MAR** [] = "MAR"
- const char **APR** [] = "APR"
- const char **MAY** [] = "MAY"
- const char **JUN** [] = "JUN"
- const char **JUL** [] = "JUL"
- const char **AUG** [] = "AUG"
- const char **SEP** [] = "SEP"
- const char **OCT** [] = "OCT"
- const char **NOV** [] = "NOV"
- const char **DEC** [] = "DEC"

Detailed Description

Header file for the Test Sorter class.

Author:

Tanner Jones

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

Version:

1.00 (23 September 2015))

Requires **TestSorter.h**

TestSorter.h File Reference

Header file for the Test Sorter class.

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

Classes

- class **TestSorter**

Detailed Description

Header file for the Test Sorter class.

Author:

Tanner Jones

Specifies all data of the Test Sorter class, along with the constructor

Version:

1.00 (23 September 2015))

Requires **TestSorter.cpp**

Index

INDEX