Project 3

Tanner Jones
1.0 Version
9/16/2015

Table of Contents

Table of contents

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

DataType Class Reference

Public Member Functions

- **DataType** ()
  Default constructor.
- **DataType** (int newPriority, char *newProcess)
  Initialization constructor.

Public Attributes

- int **priority**
- char **process** [STD_STR_LEN]

Static Public Attributes

- static const int **STD_STR_LEN** = 25

Constructor & Destructor Documentation

DataType::DataType ()

Default constructor.

Constructs empty **DataType**

Parameters:

None	

Note:

None

DataType::DataType (int *newPriority*, char * *newProcess*)

Initialization constructor.

Constructs **DataType** with data components

Parameters:

in	priority level to be loaded into **DataType**
in	process data to be loaded into **DataType**

Note:

None

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

- **DataType.h**
- **DataType.cpp**

PriorityQueue< DataType > Class Template Reference

Public Member Functions

- **PriorityQueue** ()
  Default constructor.
- **PriorityQueue** (const **PriorityQueue** &copiedQueue)
  Copy constructor.
- **~PriorityQueue** ()
  Destructor.
- bool **enqueue** (int qPriority, char qProcess[])
  enqueue(int qPriority, char qProcess [])
- bool **dequeue** (**DataType** &dataItem)
  Default ***dequeue(DataType &dataItem)***
- const **PriorityQueue** & **operator=** (const **PriorityQueue** &rhQueue)
  Default operator=.
- bool **peekAtFront** (**DataType** &dataItem)
  Default peekAtFront.
- void **showStructure** (char list)
  Default showStructure.
- bool **isEmpty** () const
  isEmpty()

Constructor & Destructor Documentation

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

Default constructor.

None

Parameters:

None	

Note:

None

template<class DataType > PriorityQueue< DataType >::PriorityQueue (const PriorityQueue< DataType > & *copiedQueue*)

Copy constructor.

Copys all data from the copiedQueue

Parameters:

None	

Note:

None

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

Destructor.

None

Parameters:

None	

Note:

None

Member Function Documentation

template<class DataType > bool PriorityQueue< DataType >::dequeue (DataType & *dataItem*)

Default **dequeue(DataType &dataItem)**

removes the first item from the list

Parameters:

takes	the first dataType from the beginning from the list and pushes all dataType down

Note:

if the size is a fourth of the capacity the capacity is cut

template<class DataType > bool PriorityQueue< DataType >::enqueue (int *qPriority*, char *qProcess*[])

enqueue(int qPriority, char qProcess [])

Adds new **DataType** to the vector

Parameters:

loops	through the vector to see where the new dataType will be placed

Note:

if the vector is full double the memory

template<class DataType > bool PriorityQueue< DataType >::isEmpty () const

isEmpty()

tests to see if the vector is empty

Parameters:

None	

Note:

None

template<class DataType > const PriorityQueue< DataType > & PriorityQueue< DataType >::operator= (const PriorityQueue< DataType > & *rhQueue*)

Default operator=.

sets the vectors equal to each other

Parameters:

None	

Note:

None

template<class DataType > bool PriorityQueue< DataType >::peekAtFront (DataType & *dataItem*)

Default peekAtFront.

looks at the first item in vector

Parameters:

takes	the first dataType for the user

Note:

if empty returns false

template<class DataType > void PriorityQueue< DataType >::showStructure (char *list*)

Default showStructure.

ShowStructure prints out the vector of all data

Parameters:

None	

Note:

None

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

- **PriorityQueue.h**
- **PriorityQueue.cpp**

SimpleVector< DataType > Class Template Reference

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 **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

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 >::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 >::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**

File Documentation

DataType.cpp File Reference

Implementation file for **DataType** class.

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

Detailed Description

Implementation file for **DataType** class.

Implements the constructor method of the **DataType** class

Version:

1.00 (07 September 2015)

Requires **DataType.h**

DataType.h File Reference

Definition file for **DataType** class.

Classes

- class **DataType**

Detailed Description

Definition file for **DataType** class.

Specifies all data of the **DataType** class, along with the constructor

Version:

1.00 (07 September 2015)

None

PA03.cpp File Reference

Driver program to exercise the **PriorityQueue** class.

```
#include <iostream>
#include <cstring>
#include "DataType.h"
#include "SimpleVector.cpp"
#include "PriorityQueue.cpp"
```

Functions

- void **ShowMenu** ()
  ShowMenu: Displays choice of commands for exercising priority queue.
- char **GetCommandInput** (char processString[], int &priority)
  GetCommandInput: Acquires command input from user.
- int **main** ()

Variables

- const int **SMALL_STR_LEN** = 25
- const bool **VERBOSE** = false
- const char **ENDLINE_CHAR** = '\n'
- const char **PERIOD** = '.'
- const int **TEST_PQ_NUM_PRIORITIES** = 12

Detailed Description

Driver program to exercise the **PriorityQueue** class.

Allows for testing all **PriorityQueue** methods in an interactive environment

Version:

1.00 (07 September 2015)

Requires **SimpleVector.cpp**, **SimpleVector.cpp**

Function Documentation

char GetCommandInput (char *processString*[], int & *priority*)

GetCommandInput: Acquires command input from user.

Command letters are unique combinations of three letters

Parameters:

None	

Note:

Clears input string, loads command letters individually using extraction operation; adds input character for display and output line for display clearance

void ShowMenu ()

ShowMenu: Displays choice of commands for exercising priority queue.

Command letters displayed indicate operations to be conducted

Parameters:

None	

Note:

None

PriorityQueue.cpp File Reference

Implementation file for priority queue class.

```
#include "PriorityQueue.h"
#include "DataType.h"
#include "SimpleVector.h"
#include <iostream>
```

Variables

- const int **ONE** = 1
- const int **ZERO** = 0
- const char **NULL_CHAR** = '\0'

Detailed Description

Implementation file for priority queue class.

Author:

Tanner Jones

Implement's all functions in the Priority Queue class, along with the constructor

Version:

1.00 (16 September 2015))

Requires **PriorityQueue.h**

PriorityQueue.h File Reference

Header file for the priority queue class.

```
#include "SimpleVector.cpp"
#include "DataType.h"
```

Classes

- class **PriorityQueue< DataType >**

Detailed Description

Header file for the priority queue class.

Author:

Tanner Jones

Specifies all data of the Priority Queue class, along with the constructor

Version:

1.00 (16 September 2015))

Requires **PriorityQueue.cpp**

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:

1.00 (30 August 2015)

Requires **SimpleVector.h**

SimpleVector.h File Reference

Definition file for **SimpleVector** class.

```
#include <stdexcept>
```

Classes

- class **SimpleVector< DataType >**

Variables

- const int **DEFAULT_CAPACITY** = 10

Detailed Description

Definition file for **SimpleVector** class.

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

Version:

1.00 (30 August 2015)

None

Index

INDEX