

Searching

0.2.0

Generated by Doxygen 1.8.17

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 myFIFO Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	5
3.1.2.1 myFIFO()	5
3.1.3 Member Function Documentation	6
3.1.3.1 add()	6
3.1.3.2 bufLen()	6
3.1.3.3 getElement()	6
3.1.3.4 lenFull()	7
3.1.3.5 printStats()	7
3.1.3.6 remove()	8
3.2 myHashSearch Class Reference	8
3.2.1 Detailed Description	8
3.2.2 Constructor & Destructor Documentation	9
3.2.2.1 myHashSearch()	9
3.2.3 Member Function Documentation	9
3.2.3.1 fillStorage()	9
3.2.3.2 printStorage()	9
3.2.3.3 search()	10
3.2.4 Member Data Documentation	10
3.2.4.1 lenStorage	10
3.2.4.2 storage	10
3.3 mySearch Class Reference	11
3.3.1 Detailed Description	11
3.3.2 Constructor & Destructor Documentation	11
3.3.2.1 mySearch()	11
3.3.3 Member Function Documentation	12
3.3.3.1 binSearch()	12
3.3.3.2 fillStorage()	12
3.3.3.3 printStorage()	13
3.3.3.4 seqSearch()	13
3.3.4 Member Data Documentation	13
3.3.4.1 storage	14
4 File Documentation	15
4.1 /home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.cpp File Reference	15

4.1.1 Detailed Description	15
4.2 /home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.h File Reference	16
4.2.1 Detailed Description	17
4.3 /home/drseth/CPTR227/20210217SearchClassDemo/src/main.cpp File Reference	17
4.3.1 Detailed Description	18
4.3.2 Function Documentation	18
4.3.2.1 avg1()	18
4.3.2.2 main()	19
4.4 /home/drseth/CPTR227/20210217SearchClassDemo/src/myHashing.cpp File Reference	19
4.4.1 Function Documentation	20
4.4.1.1 avg1()	20
4.4.1.2 main()	21
Index	23

Chapter 1

Class Index

1.1 Class List

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

myFIFO	5
myHashSearch	8
mySearch	11

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

/home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.cpp	
This is a simple implementation of a FIFO queue	15
/home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.h	
This is a simple implementation of a FIFO queue	16
/home/drseth/CPTR227/20210217SearchClassDemo/src/main.cpp	
This demonstrates header files, separate cpp files, and some searching	17
/home/drseth/CPTR227/20210217SearchClassDemo/src/myHashing.cpp	19

Chapter 3

Class Documentation

3.1 myFIFO Class Reference

```
#include <fifo.h>
```

Public Member Functions

- [myFIFO](#) ()
- bool [add](#) (int x)
- int [remove](#) ()
- void [printStats](#) ()
- int [lenFull](#) ()
- int [bufLen](#) ()
- int [getElement](#) (int ii)

3.1.1 Detailed Description

Implements an integer FIFO

Definition at line 18 of file fifo.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 myFIFO()

```
myFIFO::myFIFO ( )
```

Constructor

Definition at line 17 of file fifo.cpp.

```
17     {
18         for(int ii = 0; ii < bufLength; ii++) {
19             buffer[ii] = 0;
20         }
21     }
```

3.1.3 Member Function Documentation

3.1.3.1 add()

```
bool myFIFO::add (
    int x )
```

Adds a integer to the back of the FIFO

Parameters

x	Integer to add to the FIFO
---	----------------------------

Returns

true if successful, false otherwise

Definition at line 29 of file fifo.cpp.

```
29         {
30     //if(bufBack < bufLength) {
31     if(length < bufLength) {
32         buffer[bufBack] = x; // Add value to buffer
33         bufBack++; // equivalent to bufBack = bufBack + 1
34         bufBack = bufBack % bufLength; // wraps around to the beginning
35         length++; // increment length since an element was added
36         return(true);
37     } else {
38         cout << "bufBack exceeded buffer length" << endl;
39         return(false);
40     }
41 }
```

3.1.3.2 bufLen()

```
int myFIFO::bufLen ( )
```

Returns the length of the buffer

Definition at line 103 of file fifo.cpp.

```
103     {
104         return(bufLength);
105 }
```

3.1.3.3 getElement()

```
int myFIFO::getElement (
    int ii )
```

Returns iith element of the FIFO

Parameters

<i>ii</i>	- which element to return
-----------	---------------------------

Definition at line 112 of file fifo.cpp.

```

112         {
113     // check ii for invalid values
114     // return the iith element
115     return(buffer[(bufFront + ii) % bufLength]);
116 }
```

3.1.3.4 lenFull()

```
int myFIFO::lenFull ( )
```

Returns the number of full spaces in the fifo

Definition at line 96 of file fifo.cpp.

```

96     {
97     return(length);
98 }
```

3.1.3.5 printStats()

```
void myFIFO::printStats ( )
```

Prints the information about the buffer

Definition at line 65 of file fifo.cpp.

```

65     {
66     cout << "-----" << endl;
67     cout << "bufFront = " << bufFront << " stored at " << &bufFront << endl;
68     cout << "bufBack = " << bufBack << " stored at " << &bufBack << endl;
69     //      cout << "buffer stored at " << buffer << " is:" << endl;
70     cout << "length = " << length << endl;
71     /*
72     // print front
73     for(int ii = 0; ii < bufLength; ii++) {
74         if(ii == bufFront)
75             cout << 'f';
76         cout << '\t';
77     }
78     cout << endl;
79     for(int ii = 0; ii < bufLength; ii++) {
80         cout << buffer[ii] << '\t';
81     }
82     cout << endl;
83     for(int ii = 0; ii < bufLength; ii++) {
84         if(ii == bufBack)
85             cout << 'b';
86         cout << '\t';
87     }
88     cout << endl;
89     */
90     cout << "===== " << endl;
91 }
```

3.1.3.6 remove()

```
int myFIFO::remove ( )
```

Removes an integer from front of the FIFO

Returns

value removed from FIFO, -999999999 if error

Definition at line 48 of file fifo.cpp.

```
48     {
49     //if(bufFront < bufLength) {
50     if(length > 0) { // bufFront == bufBack means the buffer is empty
51         int retVal = buffer[bufFront];
52         bufFront++;
53         bufFront = bufFront % bufLength;
54         length--; // decrement length since an element was removed
55         return(retVal);
56     } else {
57         cout << "Error tried to remove beyond end of buffer" << endl;
58         return(-999999999);
59     }
60 }
```

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

- [/home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.h](#)
- [/home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.cpp](#)

3.2 myHashSearch Class Reference

Public Member Functions

- [myHashSearch](#) (int size)
- void [fillStorage](#) (int start)
- void [printStorage](#) ()
- int [search](#) (int searchTerm, int &N)

Public Attributes

- vector< int > [storage](#)
Variable that stores the array.
- int [lenStorage](#)

3.2.1 Detailed Description

This class implements storage and hash search examples

Definition at line 19 of file myHashing.cpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 myHashSearch()

```
myHashSearch::myHashSearch (
    int size ) [inline]
```

Constructor: Only stores values ≥ 0

Definition at line 27 of file myHashing.cpp.

```
27     {
28         cout << "Added a seqSearch instance" << endl;
29         lenStorage = size;
30         // create and initialize the storage
31         for(int ii = 0; ii < size; ii++) {
32             storage.push_back(-1);
33         }
34     }
```

3.2.3 Member Function Documentation

3.2.3.1 fillStorage()

```
void myHashSearch::fillStorage (
    int start ) [inline]
```

Fills storage with sequential numbers starting with start

Parameters

<i>start</i>	- The number to start filling at
--------------	----------------------------------

Definition at line 41 of file myHashing.cpp.

```
41     {
42         int ind; // index into the hash table
43         for(int ii = 0; ii < lenStorage; ii++) {
44             cout << "Adding " << start << " at location " << start % lenStorage << endl;
45             storage.at(start % lenStorage) = start;
46             start++;
47         }
48     }
```

3.2.3.2 printStorage()

```
void myHashSearch::printStorage ( ) [inline]
```

prints the contents of storage (beware of calling on large values)

Definition at line 53 of file myHashing.cpp.

```

53         {
54             if(lenStorage < 100) {
55                 for(int ii = 0; ii < lenStorage; ii++) {
56                     cout << storage.at(ii) << '\t';
57                 }
58                 cout << endl;
59             }
60         }

```

3.2.3.3 search()

```

int myHashSearch::search (
    int searchTerm,
    int & N ) [inline]

```

Hash search for the value passed

Parameters

<i>searchTerm</i>	The term to search for
<i>N</i>	Returns the number of iterations to find searchTerm (Pass by reference)

Returns

Returns the location of searchTerm or -1 if not found

Definition at line 69 of file myHashing.cpp.

```

69         {
70             N = 1; // initialize N
71             return(storage.at(searchTerm % lenStorage));
72         }

```

3.2.4 Member Data Documentation

3.2.4.1 lenStorage

```
int myHashSearch::lenStorage
```

Definition at line 22 of file myHashing.cpp.

3.2.4.2 storage

```
vector<int> myHashSearch::storage
```

Variable that stores the array.

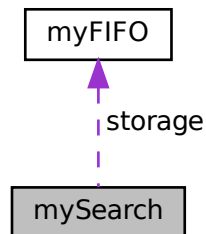
Definition at line 21 of file myHashing.cpp.

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

- </home/drseth/CPTR227/20210217SearchClassDemo/src/myHashing.cpp>

3.3 mySearch Class Reference

Collaboration diagram for mySearch:



Public Member Functions

- [mySearch](#) ()
- void [fillStorage](#) (int start)
- void [printStorage](#) ()
- int [seqSearch](#) (int searchTerm, int &N)
- int [binSearch](#) (int searchTerm, int &N)

Public Attributes

- [myFIFO storage](#)

Variable that stores the array.

3.3.1 Detailed Description

This class implements storage and search examples

Definition at line 19 of file main.cpp.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 mySearch()

```
mySearch::mySearch ( ) [inline]
```

Constructor

Definition at line 26 of file main.cpp.

```
26         {
27             cout << "Added a seqSearch instance" << endl;
28         }
```

3.3.3 Member Function Documentation

3.3.3.1 binSearch()

```
int mySearch::binSearch (
    int searchTerm,
    int & N ) [inline]
```

Binary search for the value passed

This requires the data to be ordered in increasing value. This is based off the example in Malik's Data Structures in C++ 2nd Ed.

Parameters

<i>searchTerm</i>	The term to search for
<i>N</i>	Returns the number of iterations to find searchTerm (Pass by reference)

Returns

Returns the location of searchTerm or -1 if not found

Definition at line 74 of file main.cpp.

```
74         {
75             int first = 0; // index to first value to search
76             int last = storage.lenFull() - 1; // index to last value to search
77             int mid; // index to the middle element
78             bool found = false; // true if searchTerm is found
79             N = 0; // initialize N
80
81             while((first <= last) && !found) {
82                 N++; // increment iteration counter
83                 mid = (first + last)/2;
84
85                 if(storage.getElement(mid) == searchTerm) {
86                     found = true;
87                 } else if(storage.getElement(mid) > searchTerm) {
88                     last = mid - 1;
89                 } else { // searchTerm is > mid->value
90                     first = mid + 1;
91                 }
92             }
93             if(found) {
94                 return(mid);
95             } else {
96                 return(-1);
97             }
98         }
```

3.3.3.2 fillStorage()

```
void mySearch::fillStorage (
    int start ) [inline]
```

Fills storage with sequential numbers starting with start

Parameters

<i>start</i>	- The number to start filling at
--------------	----------------------------------

Definition at line 35 of file main.cpp.

```

35         {
36             for(int ii = 0; ii < storage.bufLen(); ii++){
37                 storage.add(start++);
38             }
39         }

```

3.3.3.3 printStorage()

```
void mySearch::printStorage ( ) [inline]
```

Definition at line 41 of file main.cpp.

```

41         {
42             storage.printStats();
43         }

```

3.3.3.4 seqSearch()

```
int mySearch::seqSearch (
    int searchTerm,
    int & N ) [inline]
```

Sequential search for the value passed

Parameters

<i>searchTerm</i>	The term to search for
<i>N</i>	Returns the number of iterations to find searchTerm (Pass by reference)

Returns

Returns the location of searchTerm or -1 if not found

Definition at line 52 of file main.cpp.

```

52         {
53             N = 0; // initialize N
54             for(int ii = 0; ii < storage.lenFull(); ii++) {
55                 if(storage.getElement(ii) == searchTerm) {
56                     N = ii+1;
57                     return(ii);
58                 }
59             }
60             N = storage.lenFull();
61             return(-1);
62         }

```

3.3.4 Member Data Documentation

3.3.4.1 storage

`myFIFO mySearch::storage`

Variable that stores the array.

Definition at line 21 of file main.cpp.

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

- [/home/drseth/CPTR227/20210217SearchClassDemo/src/main.cpp](#)

Chapter 4

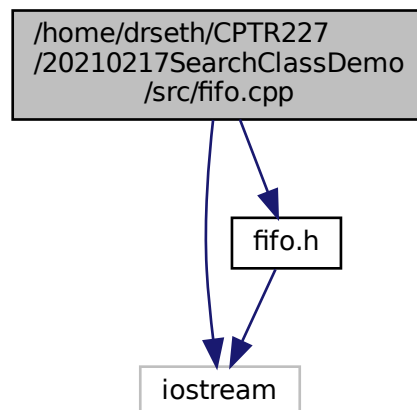
File Documentation

4.1 /home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.cpp File Reference

This is a simple implementation of a FIFO queue.

```
#include <iostream>
#include "fifo.h"
```

Include dependency graph for fifo.cpp:



4.1.1 Detailed Description

This is a simple implementation of a FIFO queue.

This only uses arrays, no STL

Author

Seth McNeill

Date

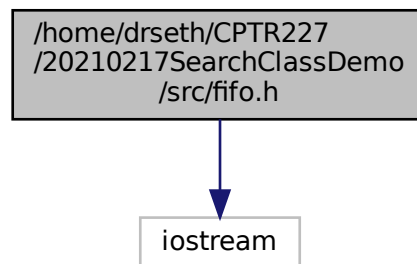
2021 February 02

4.2 `/home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.h` File Reference

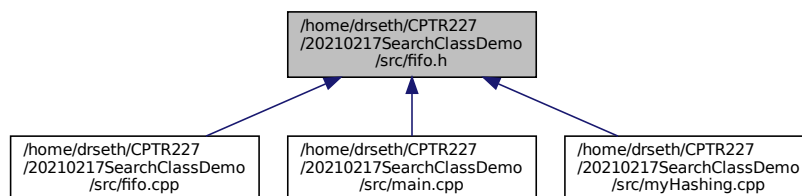
This is a simple implementation of a FIFO queue.

```
#include <iostream>
```

Include dependency graph for `fifo.h`:



This graph shows which files directly or indirectly include this file:



Classes

- class `myFIFO`

4.2.1 Detailed Description

This is a simple implementation of a FIFO queue.

This only uses arrays, no STL

Author

Seth McNeill

Date

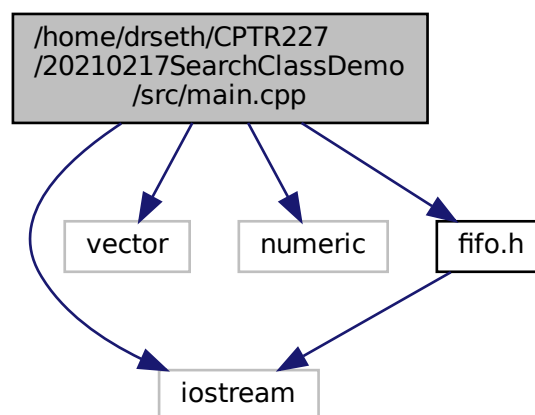
2021 February 02

4.3 /home/drseth/CPTR227/20210217SearchClassDemo/src/main.cpp File Reference

This demonstrates header files, separate cpp files, and some searching.

```
#include <iostream>
#include <vector>
#include <numeric>
#include "fifo.h"
```

Include dependency graph for main.cpp:



Classes

- class [mySearch](#)

Functions

- double `avg1` (vector< int > const &v)
- int `main` (int, char **)

4.3.1 Detailed Description

This demonstrates header files, separate cpp files, and some searching.

Implements and times sequential searching using FIFO class

Author

Seth McNeill

Date

2021 February 17

4.3.2 Function Documentation

4.3.2.1 avg1()

```
double avg1 (
    vector< int > const & v )
```

Calculate the average value of a integer vector

This is taken from: <https://stackoverflow.com/a/35833470> It uses std::accumulate.

Parameters

<code>v</code>	is a integer std::vector
----------------	--------------------------

Returns

The average value of the contents of v

Definition at line 111 of file main.cpp.

```
111         {
112     return 1.0 * accumulate(v.begin(), v.end(), 0LL) / v.size();
113 }
```

4.3.2.2 main()

```
int main (
    int ,
    char ** )
```

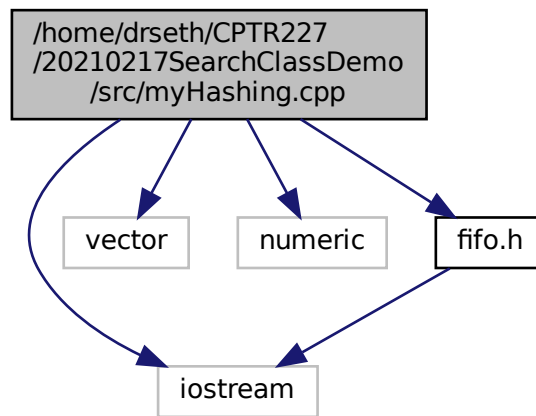
Definition at line 115 of file main.cpp.

```
115         {
116             mySearch s1;
117             int nIterations;
118             vector<int> allIters;
119             s1.fillStorage(0);
120             s1.printStorage();
121         /*
122             cout << "Sequential searching" << endl;
123             for(int ii = 0; ii < (s1.storage.lenFull()+1); ii++)
124             {
125                 //cout << "Search for " << ii << " returns ";
126                 //cout << s1.search(ii, nIterations) << " in " << nIterations;
127                 //cout << " iterations" << endl;
128                 s1.seqSearch(ii, nIterations);
129                 allIters.push_back(nIterations);
130             }
131             cout << "Calculating the average" << endl;
132             cout << "The average number of iterations for sequential search is ";
133             cout << avg1(allIters) << endl;
134         */
135             cout << "Binary searching" << endl;
136             for(int ii = 0; ii < (s1.storage.lenFull()+1); ii++)
137             {
138                 s1.binSearch(ii, nIterations);
139                 allIters.push_back(nIterations);
140             }
141             cout << "Calculating the average" << endl;
142             cout << "The average number of iterations for binary search is ";
143             cout << avg1(allIters) << endl;
144
145     }
```

4.4 /home/drseth/CPTR227/20210217SearchClassDemo/src/myHashing.cpp File Reference

```
#include <iostream>
#include <vector>
#include <numeric>
#include "fifo.h"
```

Include dependency graph for myHashing.cpp:



Classes

- class `myHashSearch`

Functions

- double `avg1` (`vector< int > const &v`)
- int `main` (`int, char **`)

4.4.1 Function Documentation

4.4.1.1 `avg1()`

```
double avg1 (  
    vector< int > const & v )
```

Calculate the average value of a integer vector

This is taken from: <https://stackoverflow.com/a/35833470> It uses `std::accumulate`.

Parameters

<code>v</code>	is a integer <code>std::vector</code>
----------------	---------------------------------------

Returns

The average value of the contents of v

Definition at line 86 of file myHashing.cpp.

```
86     {
87     return 1.0 * accumulate(v.begin(), v.end(), 0LL) / v.size();
88 }
```

4.4.1.2 main()

```
int main (
    int ,
    char ** )
```

Definition at line 90 of file myHashing.cpp.

```
90     {
91     int lenHashTable = 5;
92     myHashSearch s1(lenHashTable);
93     int nIterations;
94     vector<int> allIters;
95     s1.printStorage();
96     s1.fillStorage(11);
97     s1.printStorage();
98
99     cout << "Hash based searching" << endl;
100     for(int ii = 0; ii < (lenHashTable+1); ii++)
101     {
102         s1.search(ii, nIterations);
103         allIters.push_back(nIterations);
104     }
105     cout << "Calculating the average" << endl;
106     cout << "The average number of iterations for hash search is ";
107     cout << avg1(allIters) << endl;
108
109 }
```


Index

/home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.cpp, lenStorage, 10
15 myHashSearch, 9
/home/drseth/CPTR227/20210217SearchClassDemo/src/fifo.h, printStorage, 9
16 search, 10
/home/drseth/CPTR227/20210217SearchClassDemo/src/main.cpp, storage, 10
17 mySearch, 11
/home/drseth/CPTR227/20210217SearchClassDemo/src/myHashing.cpp, binSearch, 12
19 fillStorage, 12
mySearch, 11
printStorage, 13
seqSearch, 13
storage, 13
add
myFIFO, 6
avg1
main.cpp, 18
myHashing.cpp, 20
binSearch
mySearch, 12
bufLen
myFIFO, 6
fillStorage
myHashSearch, 9
mySearch, 12
getElement
myFIFO, 6
lenFull
myFIFO, 7
lenStorage
myHashSearch, 10
main
main.cpp, 18
myHashing.cpp, 21
main.cpp
avg1, 18
main, 18
myFIFO, 5
add, 6
bufLen, 6
getElement, 6
lenFull, 7
myFIFO, 5
printStats, 7
remove, 7
myHashing.cpp
avg1, 20
main, 21
myHashSearch, 8
fillStorage, 9
printStats
myFIFO, 7
printStorage
myHashSearch, 9
mySearch, 13
remove
myFIFO, 7
search
myHashSearch, 10
seqSearch
mySearch, 13
storage
myHashSearch, 10
mySearch, 13