NEW THINK TANK

C++ Tutorial 17

Posted by Derek Banas on May 20, 2018 in C Video Tutorial | 0 comments

In this part of my C++ tutorial I'll cover Sequence Containers which contain data that is stored in order. Previously I covered Vectors and will now cover Deques, Lists and Forward Lists.

I'm getting near the end of my core coverage of C++. Once I finish I will then move on to GUI development, Algorithms and all the other things that have been requested. All of the code follows below.

If you like videos like this consider donating $1, or simply turn off Ad Blocking software. Either helps me to keep making free tutorials for all.

Code From the Video

```
#include <cstdlib>
#include <iostream>
#include <string>
#include <vector>
#include <ctime>
#include <numeric>
#include <cmath>
#include <sstream>
#include <thread>
#include <ctime>

#include <deque>
#include <list>
#include<forward_list>

bool isEven(const int& val){
    return (val % 2) == 0;
}

int main()
{
    // ---------- SEQUENCE CONTAINERS ----------
    // Contains data stored in order

    // ---------- DEQUES ----------

    // A deque (Deck) is a dynamic array like vectors
    // except it also allows for insertion or deletion
    // from the front
    std::deque<int> deq1;

    // Add to the end and front
    deq1.push_back(5);
    deq1.push_front(1);
```

```
35
36      // Add values with assign
37      deq1.assign({11,12});
38
39      // Get the size
40      std::cout << "Size : " << deq1.size()
41              << "\n";
42
43      // Access by index
44      std::cout << deq1[0] << "\n";
45      std::cout << deq1.at(1) << "\n";
46
47      // Add at an index using an iterator
48      std::deque<int>::iterator it = deq1.begin() + 1;
49      deq1.insert(it, 3);
50
51      // Add multiple values
52      int tempArr[5] = {6,7,8,9,10};
53      deq1.insert(deq1.end(), tempArr, tempArr+5);
54
55      // Erase at an index
56      deq1.erase(deq1.end());
57
58      // Erase 1st 2 elements
59      deq1.erase(deq1.begin(), deq1.begin()+2);
60
61      // Pop first value
62      deq1.pop_front();
63
64      // Pop last
65      deq1.pop_back();
66
67      // Create a deque with 2 50s
68      std::deque<int> deq2(2,50);
69
70      // Swap values in deques
71      deq1.swap(deq2);
72
73      // Delete all values
74      deq1.clear();
75
76      // Cycle through the deque
77      for(int i : deq1)
78          std::cout << i << "\n";
79
80      // ---------- END DEQUES ----------
81
82      // ---------- LIST ----------
83      // Lists are the most efficient at inserting,
84      // moving and extracting elements, but lack
85      // direct access to elements
86
87      // Add values
88      int arr[5] = {1,2,3,4,5};
89      std::list<int> list1;
90      list1.insert(list1.begin(), arr, arr+5);
91
92      // Adding values with assign
93      list1.assign({10,20,30});
94
95      // Add to end and front
96      list1.push_back(5);
97      list1.push_front(1);
98
99      // Get the size
```

```
100     std::cout << "Size : " << list1.size()
101             << "\n";
102
103     // Can't access index
104     // std::cout << list1[0] << "\n";
105
106     // You can access the index with an iterator
107     std::list<int>::iterator it2 = list1.begin();
108     std::advance(it2, 1);
109     std::cout << "2nd Index : " <<
110             *it2 << "\n";
111
112     // Insert at an index
113     it2 = list1.begin();
114     list1.insert(it2, 8);
115
116     // Erase at an index
117     list1.erase(list1.begin());
118
119     // Erase 1st 2 elements
120     it2 = list1.begin();
121     std::list<int>::iterator it3 = list1.begin();
122     std::advance(it3, 2);
123     list1.erase(it2, it3);
124
125     // Pop first value
126     list1.pop_front();
127
128     // Pop last
129     list1.pop_back();
130
131     // Create another list
132     int arr2[6] = {10,9,8,7,6,6};
133     std::list<int> list2;
134     list2.insert(list2.begin(), arr2, arr2+5);
135
136     // Sort the list
137     list2.sort();
138
139     // Reverse the list
140     list2.reverse();
141
142     // Remove duplicates
143     list2.unique();
144
145     // Remove a value
146     list2.remove(6);
147
148     // Remove if a condition is true
149     list2.remove_if (isEven);
150
151     // Merge lists
152     list1.merge(list2);
153
154     for(int i : list2)
155         std::cout << i << "\n";
156
157     std::cout << "\n";
158
159     // Cycle through the list
160     for(int i : list1)
161         std::cout << i << "\n";
162
163     std::cout << "\n";
164
```

```
    // ---------- END LIST ----------

    // ---------- FORWARD_LIST ----------
    // A forward list is like a list, but each list
    // item only has a link to the next item in the
    // list and not to the item that proceeds it.

    // This make them the quickest of the sequence
    // containers

    std::forward_list<int> fl1;

    // Assign values
    fl1.assign({1,2,3,4});

    // Push and pop front
    fl1.push_front(0);
    fl1.pop_front();

    // Get 1st
    std::cout << "Front : " << fl1.front();

    // Get iterator for 1st element
    std::forward_list<int>::iterator it4 = fl1.begin();

    // Insert after 1st element
    it4 = fl1.insert_after(it4, 5);

    // Delete just entered 5
    it4 = fl1.erase_after(fl1.begin());

    // Place in 1st position
    fl1.emplace_front(6);

    // Remove a value
    fl1.remove(6);

    // Remove if a condition is true
    fl1.remove_if (isEven);

    std::forward_list<int> fl2;
    fl2.assign({9,8,7,6,6});

    // Remove duplicates
    fl2.unique();

    // Sort
    fl2.sort();

    // Reverse
    fl2.reverse();

    // Merge lists
    fl1.merge(fl2);

    // Clear
    fl1.clear();

    for(int i : fl1)
        std::cout << i << "\n";

    std::cout << "\n";

    for(int i : fl2)
        std::cout << i << "\n";
```

```
230
231     // ---------- END FORWARD_LIST ----------
232
233     return 0;
234 }
```

Leave a Reply

Your email address will not be published.

Comment

Name

Email

Website

Submit Comment

Search

Search

Social Networks

Facebook
YouTube
Twitter
LinkedIn

Buy me a Cup of Coffee
"Donations help me to keep the site running. One dollar is greatly appreciated." - (Pay Pal Secured)

Donate

My Facebook Page

Like Share 6K people like this. Be the first of your friends.

Archives

- October 2018
- September 2018
- August 2018
- July 2018
- June 2018
- May 2018
- April 2018
- March 2018

- February 2018
- January 2018
- December 2017
- November 2017
- October 2017
- September 2017
- August 2017
- July 2017
- June 2017
- May 2017
- April 2017
- March 2017
- February 2017
- January 2017
- December 2016
- November 2016
- October 2016
- September 2016
- August 2016
- July 2016
- June 2016
- May 2016
- April 2016
- March 2016
- February 2016
- January 2016
- December 2015
- November 2015
- October 2015
- September 2015
- August 2015
- July 2015
- June 2015
- May 2015
- April 2015
- March 2015
- February 2015
- January 2015
- December 2014
- November 2014
- October 2014
- September 2014
- August 2014
- July 2014
- June 2014
- May 2014
- April 2014
- March 2014
- February 2014
- January 2014
- December 2013
- November 2013
- October 2013
- September 2013

- August 2013
- July 2013
- June 2013
- May 2013
- April 2013
- March 2013
- February 2013
- January 2013
- December 2012
- November 2012
- October 2012
- September 2012
- August 2012
- July 2012
- June 2012
- May 2012
- April 2012
- March 2012
- February 2012
- January 2012
- December 2011
- November 2011
- October 2011
- September 2011
- August 2011
- July 2011
- June 2011
- May 2011
- April 2011
- March 2011
- February 2011
- January 2011
- December 2010
- November 2010
- October 2010
- September 2010
- August 2010
- July 2010
- June 2010
- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009