Vectors and algorithms

Anna Simon

October 7, 2015

Vectors

- Vectors are sequence containers representing arrays that can change in size.
- Use contiguous storage locations for their elements, which means that their elements can also be accessed using offsets on regular pointers to its elements
- Internally, vectors use a dynamically allocated array to store their elements. This array may need to be reallocated in order to grow in size when new elements are inserted. This is a relatively expensive task in terms of processing time, and thus, vectors do not reallocate each time an element is added to the container.
- Vector containers may allocate some extra storage to accommodate for possible growth, and thus the container may have an actual capacity greater than the storage strictly needed to contain its elements
- Compared to arrays, vectors consume more memory in exchange for the ability to manage storage and grow dynamically in an efficient way.

How to use vectors

- `#include <vector>`
- C++ provides a class of vectors: to construct, destruct and operate on a vector
- The class provides a list of functions that can be used with vectors

NOTE: the learning curve for using vectors might be a bit steep, but in a long run it is worth the effort.

Vector features

- Size changes as needed
- Append elements at the end of the vector (`push_back()`)
- Remove the last element of the vector (`pop_back()`)
- Insert a new element anywhere inside a vector (`insert()`)
- Can erase any element of the vector (`erase()`)
- "Scan" the vector forward and reverse using iterator and reverse iterator

Vector constructor

```
1 vector<float> vector1; //default constructor, takes no arguments
2
3 vector<float> vector2(vector1); //creates a vector that is an exact
      copy of vector1
4
5 vector<int> vector3(3,11); //creates a 3-element vector, each
      element is assign a value of 11
6
7 vector<int> vector4(iter1, iter2); //creates a vector by copying
      elements of another vector between the two iterators: iter1,
      iter2
```

Iterator

Vectors can be accessed by iterators: special pointers that allow for scanning the vector.

```
1     vector<int> v1;
2     for(int i=0;i<10;i++) v1.push_back(i); //appends value of i to
          the end of the vector
3     //declare two iterators pointing to the first element of v1
4     vector<int>::iterator iter1=v1.begin();
5     vector<int>::iterator iter2= v1.begin();
6
7     //scan through the v1 vector
8     for(int k=0;k<v1.size();k++){
9         if (v1[k] < 3) iter1++;
10        if (v1[k] < 7) iter2++;
11    }
12    //declare and initialize a new vector
13    vector<int> v2(iter1,iter2);
14
15    //print out the elements of the new vector
16    for(int i=0;i<v2.size();i++){
17        cout << v2[i] << " ";
18    }
19    cout << endl;
```

Algorithm

- The header `<algorithm>` defines a collection of functions especially designed to be used on ranges of elements.
- A range is any sequence of objects that can be accessed through iterators or pointers, such as an array or vectors.
- Algorithms operate through iterators directly on the values, not affecting in any way the structure of any possible container (it never affects the size or storage allocation of the container).
- Algorithms can sort elements of a container, find max/min value, count elements that meet a certain criteria, etc.

Problems

Exit pass: Complete problem 1 from the list.
Problem 2 is this week's homework.