CS 11 Data Structures and Algorithms

Assignment 9: STL and Iterators

Return to Course Homepage

Assignment 9.1

```
/*
 Name:
 Date:
 Assignment Number: 9.1
 Instructor: Dave Harden
 File: a9.1.cpp

 This program demonstrates the use of a vector and iterators to navigate and
 access the vector.  The program models capturing a set of records consisting
 of a name and a score.   The vector elements are structs.  The struct consists
 of a pointer to a C string to store a name, and an int to store a score.

 The program begins by asking the user how many scores will be recorded.
 Then the vector is instantiated with the required size, and functions are
 called to input the scores, sort them, and then display them.

 */

#include <iostream>
#include <vector>
using namespace std;

struct highscore{
    char* name;
    int score;
};

const int MAX_NAMESIZE = 24;

void getVectorSize(int& size);
void initializeData(vector<highscore>& scores);
void sortData(vector<highscore>& scores);
vector<highscore>::iterator locationOfLargest(const vector<highscore> &scores,
                                    vector<highscore>::iterator start);
void displayData(const vector<highscore>& scores);

int main()
{
    int size;
    getVectorSize(size);
    vector<highscore> scores(size);

    initializeData(scores);
    sortData(scores);
    displayData(scores);

    for (vector<highscore>::iterator i = scores.begin(); i < scores.end(); ++i){
        delete [] i->name;
    }

    return 0;
}





/*
 void getVectorSize(int& size);

 The number of scores to be entered is captured in the
 pass-by-reference parameter, size.

 */
void getVectorSize(int& size){
    cout << "How many scores will you enter?: ";
```

This study resource was shared via CourseHero.com

```
    cin >> size;
    cin.get();
}




/*
 void initializeData(vector<highscore>& scores);

 Names and scores are input to the vector, scores.

 */
void initializeData(vector<highscore>& scores)
{
    char temp[MAX_NAMESIZE];
    int counter = 1;
    vector<highscore>::iterator it;

    for(it = scores.begin(); it < scores.end(); ++it)
    {
        cout << "Enter the name for score #" << counter << ": ";
        cin.getline(temp,MAX_NAMESIZE,'\n');
        it->name = new char[strlen(temp) + 1];
        strcpy(it->name, temp);

        cout << "Enter the score for score #" << counter << ": ";
        cin >> it->score;
        cin.get();
        ++counter;
    }
    cout << endl;
}




/*
 void sortData(vector<highscore>& scores);

 The vector, scores, is sorted in descending order by score
 using a selection sort.

 */
void sortData(vector<highscore> &scores) {

    vector<highscore>::iterator it;

    for (it = scores.begin(); it != scores.end(); ++it){
        swap(*locationOfLargest(scores, it), *it);
    }
}




/*
 vector<highscore>::iterator locationOfLargest(const vector<highscore> &scores,
                                        vector<highscore>::iterator locLargest);

 This function returns an iterator pointing to the element with the largest
 score in the scores vector, in the section from the element indicated by
 the locLargest parameter, to the end of the vector.  The largest score is assumed
 to be at the beginning of the range as the function begins, and its location is
 updated as the function traverses the vector.

 */
vector<highscore>::iterator locationOfLargest(const vector<highscore> &scores,
                                          vector<highscore>::iterator locLargest){

    vector<highscore>::iterator next;
    for (next = locLargest + 1; next != scores.end(); ++next){
        if (next->score > locLargest->score){
            locLargest = next;
        }
    }
    return locLargest;
}
```

This study resource was shared via CourseHero.com

```
/*
 void displayData(const vector<highscore>& scores);

 Outputs to the console the names and scores stored in the
 vector, scores.

 */
void displayData(const vector<highscore>& scores)
{
    vector<highscore>::const_iterator it;
    cout << "Top Scorers: " << endl;

    for (it = scores.begin(); it < scores.end(); ++it)
    {
        cout << it->name << ": " << it->score << endl;
    }
}




// This version uses the range-based for loop, a C++11 feature.
// "auto" means the type is set by the context.  Here the
// type is highscore, the type of the vector element.  You
// could read the for loop header as, "For each element i in
// scores".
//
//void displayData(const vector<highscore>& scores)
//{
//    cout << "Top Scorers: " << endl;
//    for (const auto & i: scores){
//        cout << i.name << ": " << i.score << endl;
//    }
//
//}

//--------------------------------------------------------------------------
//
//  This version uses the STL sort

/*
 Name:
 Date:
 Assignment Number: 10.1
 Instructor: Dave Harden
 File: a10.cpp

 This program demonstrates the use of a vector and iterators to navigate and
 access the vector.  The program models capturing a set of records consisting
 of a name and a score.   The vector elements are structs.  The struct consists
 of a pointer to a C string to store a name, and an int to store a score.

 The program begins by asking the user how many scores will be recorded.
 Then the vector is instantiated with the required size, and functions are
 called to input the scores, sort them, and then display them.

*/

#include <iostream>
#include <vector>
using namespace std;

struct highscore{
    char* name;
    int score;
};

const int MAX_NAMESIZE = 24;

void getVectorSize(int& size);
void initializeData(vector<highscore>& scores);
void displayData(const vector<highscore>& scores);
bool sortByScore(const highscore& lhs, const highscore& rhs);  // this function is the third argument
                                                               // in the call to the STL sort function.

int main()
{
    int size;
    getVectorSize(size);
    vector<highscore> scores(size);
```

This study resource was shared via CourseHero.com

```
    initializeData(scores);
    sort(scores.begin(), scores.end(), sortByScore);  // call to STL sort.
                                                      // the third argument is a bool
                                                      // function.  Notice that sortByScore's
                                                      // arguments are omitted in the call to sort.
    displayData(scores);

    for (vector<highscore>::iterator i = scores.begin(); i < scores.end(); ++i){
        delete [] i->name;
    }

    return 0;
}






/*
 void getVectorSize(int& size);

 The number of scores to be entered is captured in the
 pass-by-reference parameter, size.

 */
void getVectorSize(int& size){
    cout << "How many scores will you enter?: ";
    cin >> size;
    cin.get();
}






/*
 void initializeData(vector<highscore>& scores);

 Names and scores are input to the vector, scores.

 */
void initializeData(vector<highscore>& scores)
{
    char temp[MAX_NAMESIZE];
    int counter = 1;
    vector<highscore>::iterator it;

    for(it = scores.begin(); it < scores.end(); ++it)
    {
        cout << "Enter the name for score #" << counter << ": ";
        cin.getline(temp,MAX_NAMESIZE,'\n');
        it->name = new char[strlen(temp) + 1];
        strcpy(it->name, temp);

        cout << "Enter the score for score #" << counter << ": ";
        cin >> it->score;
        cin.get();
        ++counter;
    }
    cout << endl;
}






/*
 bool sortByScore(const highscore &lhs, const highscore &rhs);

 Provides the sort algorithm, which here is a descending sort
 by the data member, score, of the highscore struct. lhs and
 rhs are two highscore structs to be compared.

*/
bool sortByScore(const highscore &lhs, const highscore &rhs) {
    return lhs.score > rhs.score;
}






/*
 void displayData(const vector<highscore>& scores);
```

This study resource was shared via CourseHero.com

```
 Outputs to the console the names and scores stored in the
 vector, scores.

 */
void displayData(const vector<highscore>& scores)
{
    vector<highscore>::const_iterator it;
    cout << "Top Scorers: " << endl;

    for (it = scores.begin(); it < scores.end(); ++it)
    {
        cout << it->name << ": " << it->score << endl;
    }
}
```

© 2010 - 2016 Dave Harden

This study resource was shared via CourseHero.com