Example of using Vectors and Iterators STL

Posted by Alejandro G. Carlstein Ramos Mejia on October 15, 2010 October 15, 2010 About Programming / C++

Example of using vectors and iterators STL.

NOTIFICATION: These examples are provided for educational purposes. Using this code is under your own responsibility and risk. The code is given 'as is'. I do not take responsibilities of how they are used.

CardHolder.cc:

```
#ifndef CARDSHOLDER_CC
#define CARDSHOLDER_CC

#include <iostream>

using namespace std;

template <typename T, int N>

class CardsHolder {

 private:

 public:

 insert(N value);

 CardsHolder(void);

 ~CardsHolder(void);
};

CardsHolder::CardsHolder(void){
 cout << endl << '[CARDHOLDER]' << endl;
}

CardsHolder::~CardsHolder(void){
 cout << endl << '[~CARDHOLDER]' << endl;
}

#endif
```

CardHolder.h:

```
#ifndef CARDSHOLDER
#define CARDSHOLDER

#include <iostream>

using namespace std;

class CardsHolder {

 private:

 public:

 CardsHolder(void);

 insert(int value);

 ~CardsHolder(void);
};

CardsHolder::CardsHolder(void){
 cout << endl << '[CARDHOLDER]' << endl;
}

CardsHolder::insert(int value){
}

CardsHolder::~CardsHolder(void){
 cout << endl << '[~CARDHOLDER]' << endl;
}

#endif
```

mainDriver.cpp:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <time.h>

using namespace std;

template <typename T>
void display(const T& c);

template <typename Iterator, typename T>
void display(Iterator first, Iterator last, const T& c);

/**
 * The purpose of this step is to isolate any problems with
 * simply including and using the appropriate parts of the STL.
 * Write a program that creates a single instance of
 * the container you selected, one that holds integers.
```

```
 */
int main(int argc, char* argv[]){

 // Your program should contain a single line...
 // one that instantiates the templated container for integers.
 // ie: vector<int> my_vect
 vector<int> cards_vector;

 // Insert integers 1-52 into your container.
 for (int i = 1; i < 53; i++)
  cards_vector.push_back(i);

 // Print them (using display() - note that this does not use iterators.
 // Write one using iterators instead if you want to!)
 display(cards_vector);

 cout << endl << '-----------------------------------------------'<< endl;

 // Shuffle the integers (using random_shuffle()).
 srand(time(NULL));
 random_shuffle(cards_vector.begin(), cards_vector.end());

 // Display them again, confirming that they are shuffled.
 display(cards_vector);

 // In a loop that is terminated by the STL container
 // becoming empty ( not because you've done it 52 times),
 // extract the integers from the front of the container,
 // printing them out one at a time.

 return 0;
}

/**
 * Function template to display elements of any type
 * (for which the output operator is defined) stored in
 * a container c (for which [] and size() are defined).
 *
 * Precondition:  Container is a type parameter.
 *
 * Postcondition: Values stored in c are output to out.
 */
template <typename T>
void display(const T& c){

 for (unsigned int i = 0; i < c.size(); i++)
    cout << c[i] << '  ';

 cout << endl;

}

template <typename Iterator, typename T>
```

```
void display(Iterator first, Iterator last, const T& c) {
 while (first != last){
      cout << c[first] << ' ';
      ++first;
    }
}
```

If you encounter any problems or errors, please let me know by providing an example of the code, input, output, and an explanation. Thanks.