bool empty() const;
   // returns true iff size() == 0;

void assign(unsigned n, const T & x=T());
   // clears this vector and then inserts n copies of the element x;
   // precondition: n >= 0;
   // postcondition: size() == n;

T & operator[](unsigned i);
   // returns element number i;
   // precondition: 0 <= i < size();
   // result is unpredictable if precondition is false;

T & at(unsigned i);
   // returns element number i;
   // precondition: 0 <= i < size();
   // exception is thrown if precondition is false;

T & front();
   // returns the first element of this vector;

T & back();
   // returns the last element of this vector;

iterator begin();
   // returns an iterator pointing to the first element of this vector;

iterator end();
   // returns an iterator pointing to the dummy element that follows
   // the last element of this vector;

reverse_iterator rbegin();
   // returns a reverse iterator pointing to the last element of this vector;

reverse_iterator rend();
   // returns a reverse iterator pointing to the dummy element that precedes
   // the first element of this vector;

void push_back(const T & x);
   // appends a copy of the element x to the back of this vector;
   // postcondition: back() == x;
   // postcondition: size() has been incremented;

void pop_back();
   // removes the last element of this vector;
   // precondition: size() > 0;
   // postcondition: size() has been decremented;
iterator insert(iterator p, const T& x);
// inserts a copy of the element x at position p; returns p;
// precondition: begin() <= p <= end();
// postcondition: size() has been incremented;

iterator erase(iterator p);
// removes the element at position p; returns p
// precondition: begin() <= p <= end();
// postcondition: size() has been decremented;

iterator erase(iterator p1, iterator p2);
// removes the elements from position p1 to the position before p2;
// returns p1;
// precondition: begin() <= p1 <= p2 <= end();
// postcondition: size() has been decreased by int(p2-p1);

void clear();
// removes all the elements from this vector;
// postcondition: size() == 0;

EXAMPLE D.1 Using an Iterator on a vector Object

```cpp
#include <iostream>
#include <vector>
using namespace std;
typedef vector<int>::iterator It;

int main()
{
    vector<int> v(4);
    for (int i=0; i<4; i++)
        v[i] = 222*i + 333;
    cout << "Using the iterator it in a for loop:\n";
    for (It it=v.begin(); it!=v.end(); it++)
        cout << "*it=" << *it << "\n";
    cout << "Using the iterator p in a while loop:\n";
    It p=v.begin();
    while (p!=v.end())
        cout << "*p++=" << *p++ << "\n";
}
```

The vector v has 4 elements: 333, 555, 777, and 999. The second for loop uses the iterator it to traverse the vector v from beginning to end, accessing each of its elements with *it. The while loop has the same effect using *p.