```cpp
//      orig.: Ex630.cpp
//  extended to the use of iterators

//      Sec. 6.1.3 of lecture
//      Pointers and arrays, Iteratoren
//  requires compiler option  -std=c++11

#include <algorithm>                    // sort as general algorithm
#include <iostream>
#include <list>                         // list; sort for list
#include <vector>                       // vector
using namespace std;

int main()
{
    {
        const int N=10;
        double    x[N];                 // static C-array

        double* const px = &x[0];       // px and pp and x point to the same address
        double *pp = x;

        if ( px == pp)
        {
            cout << endl << " px and pp are identical" << endl;
        }

//              initialize x
        for (int i = 0; i < N; ++i )
        {
            x[i]    = (i+1)*(i+1);
//            *(px+i) = (i+1)*(i+1);            // x[i] = ... in pointer notation
        }

//              check element 6
        int i = 6;
        cout << endl;
        cout << x[i]    << endl;
        cout << *(x+i)  << endl;
        cout << px[i]   << endl;
        cout << *(px+i) << endl << endl;

//              output of vector x;                 // pointer pi as loop variable
//      x+N;                                    // pointer to nearest address  a f t e r  last element
        for (double* pi =x; pi !=x+N; ++pi)
        {
            cout << "  " << *pi << endl;
        }
    }
    cout << "\n###############################\n";

//  and now with a C++-vector
    {
        const int N=10;
        vector<double> x(N);
        for (size_t i = 0; i < x.size(); ++i )
        {
            x[i] = -(i+1)*(i+1);                // x[i] = ...
        }

        sort(x.begin(),x.end());       //  sort ascending; general algorithm sort(); Aufsteigend anordnen

        // iterator
        vector<double>::iterator pi;
        for (pi=x.begin(); pi!=x.end(); ++pi)
        {
            cout << "  " << *pi << endl;
        }

    }

    cout << "\n###############################\n";

//  now using list | und jetzt mit Liste
//  list has no random access, i.e., no index operator []  or  at() is available
//                 |  Bei list ist kein wahlfreier Zugriff mehr moeglich, d.h. kein [], at()
    {
        const int N=10;
        list<double> x(N);
        int i=0;
        for (list<double>::iterator pi=x.begin(); pi!=x.end(); ++pi)
        {
            *pi = -(i+1)*(i+1);         // x[i] = ...
            ++i;
        }

        // iterator (auto requires Compiler option:  -std=c++11)
        for (auto pi=x.begin(); pi!=x.end(); ++pi)
        {
            cout << "  " << *pi << endl;
        }

        x.sort();        //  sort ascending; special methods sort() for list
        cout << "\n------------------------\n";
        // even more compact by using Range-FOR
        for (auto pi : x)   // Range-FOR         (-std=c++11)
        {
            cout << " # " << pi << endl;
        }

    }
    return 0;
}



```