

```

1: //      orig.: Ex630.cpp
2: //      extended to the use of iterators
3:
4: //      Sec. 6.1.3 of lecture
5: //      Pointers and arrays, Iteratoren
6: //      requires compiler option -std=c++11
7:
8: #include <algorithm>           // sort as general algorithm
9: #include <iostream>
10: #include <list>               // list; sort for list
11: #include <vector>             // vector
12: using namespace std;
13:
14: int main()
15: {
16:     {
17:         const int N=10;
18:         double    x[N];           // static C-array
19:
20:         double* const px = &x[0]; // px and pp and x point to t
he same address
21:         double *pp = x;
22:
23:         if ( px == pp)
24:         {
25:             cout << endl << " px and pp are identical" << endl;
26:         }
27:
28:         //      initialize x
29:         for (int i = 0; i < N; ++i )
30:         {
31:             x[i]      = (i+1)*(i+1);
32:             *(px+i) = (i+1)*(i+1); // x[i] = ... in poin
ter notation
33:         }
34:
35:         //      check element 6
36:         int i = 6;
37:         cout << endl;
38:         cout << x[i]      << endl;
39:         cout << *(x+i)    << endl;
40:         cout << px[i]     << endl;
41:         cout << *(px+i)  << endl << endl;
42:
43:         //      output of vector x;           // pointer pi as
loop variable
44:         //      x+N;                         // pointer to nearest
address a f t e r last element
45:         for (double* pi =x; pi !=x+N; ++pi)
46:         {
47:             cout << "    " << *pi << endl;
48:         }
49:     }
50:     cout << "\n#####\n";
51:
52:     // and now with a C++-vector
53:     {

```

```

54:         const int N=10;
55:         vector<double> x(N);
56:         for (size_t i = 0; i < x.size(); ++i )
57:         {
58:             x[i] = -(i+1)*(i+1);           // x[i] = ...
59:         }
60:
61:         sort(x.begin(),x.end());           // sort ascending; general al
gorithm sort(); Aufsteigend anordnen
62:
63:         // iterator
64:         vector<double>::iterator pi;
65:         for (pi=x.begin(); pi!=x.end(); ++pi)
66:         {
67:             cout << "  " << *pi << endl;
68:         }
69:
70:     }
71:
72:     cout << "\n#####\n";
73:
74:     // now using list | und jetzt mit Liste
75:     // list has no random access, i.e., no index operator [] or at()
is available
76:     // | Bei list ist kein wahlfreier Zugriff mehr moegl
ich, d.h. kein [], at()
77:     {
78:         const int N=10;
79:         list<double> x(N);
80:         int i=0;
81:         for (list<double>::iterator pi=x.begin(); pi!=x.end(); ++pi)
82:         {
83:             *pi = -(i+1)*(i+1);           // x[i] = ...
84:             ++i;
85:         }
86:
87:         // iterator (auto requires Compiler option: -std=c++11)
88:         for (auto pi=x.begin(); pi!=x.end(); ++pi)
89:         {
90:             cout << "  " << *pi << endl;
91:         }
92:
93:         x.sort();           // sort ascending; special methods sort() f
or list
94:         cout << "\n-----\n";
95:         // even more compact by using Range-FOR
96:         for (auto pi : x) // Range-FOR (-std=c++11)
97:         {
98:             cout << " # " << pi << endl;
99:         }
100:
101:     }
102:     return 0;
103: }
104:
105:
106:

```

./main.cpp

Tue Feb 23 17:00:13 2021

3

107:

108: