```
    1: #include <algorithm>      // sort
    2: #include <iomanip>        // setw
    3: #include <iostream>
    4: #include <numeric>        // iota
    5: #include <vector>
    6: using namespace std;
    7:
    8: //! @brief Determines the permutation vector for ascending order of a vector.
    9: //!
   10: //! @param[in] v  vector to determine sorting indices
   11: //! @return permutation order
   12: //! @see Ideas in <a href="http://stackoverflow.com/questions/1577475/c-sorting-and-keeping-track-
of-indexes">Stack Overflow</a>
   13: //!
   14: template <typename T>
   15: vector<size_t> sort_indexes(const vector<T> &v);
   16:
   17: vector<size_t> inverse_indexes(const vector<size_t> &v);
   18:
   19: template <class T>
   20: ostream& operator<<(ostream &s, const vector<T> &idx);
   21:
   22:
   23: int main()
   24: {
   25:     cout << "Hello world!" << endl;
   26:
   27:     //const vector<double> v {1.23, -4.56, -6.7, 2.3, 1.1};        // initial
   28:     const vector<double> v {3.1,2.1,12.1,9.1,8.1,3.1,7.1,6.1};
   29:     cout << "   Orig. vector: " << v << endl;
   30:     const vector<size_t>  idx =  sort_indexes(v);
   31:
   32:     cout << "   Sort   index: " << idx << endl;
   33:
   34: // -----------  Permute the original vector using the index vector ---------
   35:     vector<double> sv(v.size());
   36:                                 // Version a)  via Loop
   37: //    for (size_t k=0; k<idx.size(); ++k)
   38: //    {
   39: //        sv[k] = v[ idx[k] ];
   40: //    }
```

Erhalte Indexvektor bzgl. Umsortierung.

Sortierter Vektor

```
41:                                 // Version b)  via "transform" and Lambda
42:     transform(idx.begin(),idx.end(), sv.begin(),
43:               [&v](size_t k) { return v[k]; }
44:               );
45:     cout << "Permuted vector: " << sv << endl;
46:
47:     //
48:     auto inv_idx = inverse_indexes(idx);
49:     cout << endl << "inv. sort index: " << inv_idx << endl;
50:
51:
52:     return 0;
53: }
54:
55:
56:
57: template <typename T>
58: vector<size_t> sort_indexes(const vector<T> &v)
59: {
60:     // initialize original index locations
61:     vector<size_t> idx(v.size());
62:     //for (size_t i = 0; i != idx.size(); ++i) idx[i] = i;  // per loop
63:     iota(begin(idx),end(idx),0);
64:
65:     // sort indexes based on comparing values in v
66:     sort(idx.begin(), idx.end(),
67:          [&v](size_t i1, size_t i2) -> bool
68:           { return v[i1] < v[i2]; }
69:         );
70:
71:     return idx;
72: }
73:
74: vector<size_t> inverse_indexes(const vector<size_t> &idx)
75: {
76:     return sort_indexes(idx);
77: }
78:
79:
80: template <class T>
81: ostream& operator<<(ostream &s, const vector<T> &v)
```

Init. Index-Vektor [0, 1, ...]

Sortieren der Indizes

auch andere binäre Vergleichsfkt. möglich

```
82: {
83:     for (auto it: v) { cout << "  " << setw(5) << it; }
84:     return s;
85: }
```