Algorithms and Data Structures / 15

Types of containers in STL library (C++)

Andrzej Pisarski

Plan of lecture

Components of STL library

Standard Template Library (STL) – is a type of standard library that allows managing data collections using algorithms. Using the *STL* library does not require knowledge of how data structures work.

The most important components of the *STL* library are:

- **Containers** - are used to manage collections of objects of a specific type.
- **Iterators** – are used to "walk" through the elements of a collection.
- **Algorithms** – are used to process elements of a collection (searching, sorting, modifying).

Sequential containers

Containers (*container class*) manage collections of items. There are two basic types of containers: *sequential containers* and *associative containers*.

Sequential containers are ordered collections in which each element has a fixed position. This position depends on the time and place of insertion, but is independent of the element's value. The *STL* library contains three predefined sequential container classes: *vector*, *deque*, and *list*.

vector

```
1  // (C) Copyright Nicolai M. Josuttis 1999.
2  #include <iostream>
3  #include <vector>
4  using namespace std;
5
6  int main()
7  {
8      vector<int> coll; // kontener wektorowy
9      // dolacz elementy o wartosciach od 1 do 6
10     for (int i=1; i<=6; ++i) {
11         coll.push_back(i);
12     }
13     // wyswietl elementy rozdzielone spacjami
14     for (int i=0; i<coll.size(); ++i) {
15         cout << coll[i] << ' ';
16     }
17     cout << endl;
18 }
```

deque (double-ended queue)

The name of the queue *deque* (pronounced "*dek*") is short for *double-ended queue*. It is a dynamic array whose size can grow on both sides of the array. This makes adding new elements to the end and to the beginning of the array very fast.

```
1  // (C) Copyright Nicolai M. Josuttis 1999.
2  #include <iostream>
3  #include <deque>
4  using namespace std;
5
6  int main()
7  {
8      deque<float> coll; // kontener deque
9      // wstawia elementy na poczatek kontenera
10     for (int i=1; i<=6; ++i) {
11         coll.push_front(i*1.1);
12     }
13
14     // wyswietla wszystkie elementy
15     for (int i=0; i<coll.size(); ++i) {
16         cout << coll[i] << ' ';
17     }
18     cout << endl;
19 }
```

list (double linked list)

The container *list* is implemented as a double-linked list. The advantage of a list is the fast insertion and removal of list elements regardless of the element's location.

list (double linked list)

```
1  // (C) Copyright Nicolai M. Josuttis 1999.
2  #include <iostream>
3  #include <list>
4  using namespace std;
5
6  int main()
7  {
8      list<char> coll;  // kontener list
9      // dolacz elementy od 'a' do 'z'
10     for (char c='a'; c<='z'; ++c) {
11         coll.push_back(c);
12     }
13     // wypisz i usun pierwszy element
14     while (! coll.empty()) {
15         cout << coll.front() << ' ';  // drukuj
16         coll.pop_front(); // usun p. e.
17     }
18     cout << endl;
19 }
```

Associative containers

Associative containers are collections with sorted elements. The current position of an element depends on its value according to a specified sorting criterion. The *STL* library contains the following predefined associative container classes: *set*, *multiset*, *map*, and *multimap*.

set

The containers *set* and *multiset* sort their elements automatically.
Difference: *multiset* allows to repeate elements, *set* does not.

```
1  // (C) Copyright Nicolai M. Josuttis 1999.
2  #include <iostream>
3  #include <set>
4  #include <iterator>
5  using namespace std;
6
7  int main(){/* typ kolekcji:
8     * - brak powtorzen
9     * - elementy sa wartosciami calkowitymi
10    * - porzadek malejacy */
11    typedef set<int,greater<int> > IntSet;
12    IntSet coll1;          // pusty zbior
13    // verte
```

set

```
14      // wstaw elementy w przypadkowej
         ↪  kolejnosci
15      coll1.insert(4);
16      coll1.insert(3);
17      coll1.insert(5);
18      coll1.insert(1);
19      coll1.insert(6);
20      coll1.insert(2);
21      coll1.insert(5);
22      // iteruj po wszystkich elementach i
         ↪  wypisz je
23      IntSet::iterator pos;
24      for (pos = coll1.begin(); pos !=
         ↪  coll1.end(); ++pos) {
25         cout << *pos << ' ';
26      }
27      cout << endl;
28      // verte
```

set

```
29      // wstaw ponownie wartosc 4 i przetworz
         ↪  wartosc zwracana
30      pair<IntSet::iterator,bool> status =
         ↪  coll1.insert(4);
31      if (status.second) {
32          cout << "wartosc 4 wstawiono jako
             ↪  element "
33          << distance(coll1.begin(),status.first)
             ↪  + 1 << endl;
34      }
35      else {
36          cout << "item 4 already exists" <<
             ↪  endl;
37      }// przypisz elementy do innego zbioru o
         ↪  porzadku rosnacym
38      set<int> coll2(coll1.begin(),
         ↪  coll1.end());
39      // verte
```

set

```
40      // wypisz wszystkie elementy kopii
41      copy (coll2.begin(), coll2.end(),
42      ostream_iterator<int>(cout," "));
43      cout << endl;
44      // usun wszystkie elementy az do elementu
         ↪  o wartosci 3
45      coll2.erase (coll2.begin(),
         ↪  coll2.find(3));
46      // usun wszystkie elementy o wartosci 5
47      int num; // How many elements has been
48      num = coll2.erase (5); // removed.
49      cout << num << " item(s) removed" <<
         ↪  endl;
50      // wypisz wszystkie elementy
51      copy (coll2.begin(), coll2.end(),
52      ostream_iterator<int>(cout," "));
53      cout << endl;
54  }
```

set

Output:

```
6 5 4 3 2 1
item 4 already exists
1 2 3 4 5 6
1 item(s) removed
3 4 6
```

Bibliography

Source of materials used in the presentation:

1. Steve Oualline - "Język C. Programowanie", Helion 2003,
2. Nicolai M. Josuttis - "C++ Biblioteka standardowa. Podręcznik programisty", Helion 2003,
3. Nicolai M. Josuttis - "*The C++ Standard Library - A Tutorial and Reference, 2nd Edition*", Addison Wesley Longman 2012.
   - http://www.cppstdlib.com/

Thank you for your attention!

- https://www.youtube.com/watch?v=UeG48Dl2sIc&t=754s