Programowanie C++

Wykład 6 (03.04.2017)

Kontenery STL

```
vector<int> t(20);

for(int i =0 ; i<20 ; i++)
{
   t[i] = i+1;
}
```

```
double max(double a, double b) {
      if(a > b)
            return a;
        else
            return b;
}

int max(int a, int b) {
      if(a > b)
            return a;
        else
            return b;
}

short max(short a, short b) {
      if(a > b)
            return a;
        else
            return b;
}
```

```
...
cout << max(6, 9);
...
```

```
template <typename Type>
Type max(Type a, Type b) {
      if(a > b)
            return a;
      else
            return b;
}
```

```
      ...
      cout << max<int>(6, 9);
      ...
```

Kontenery STL

```
#include <forward_list>
#include <iostream>
using namespace std;
int main() {

  forward_list<int> lista;
  lista.push_front(5);
  lista.push_front(3);
  lista.push_front(1);

  cout << lista.pop_front() << "\n";

  if(lista.empty())
        cout << "Lista jest pusta\n";

  return 0;
}
```

Kontenery STL

```
#include <list>
#include <iostream>
using namespace std;
int main() {

  list<int> lista;
  lista.push_front(5);
  lista.push_front(3);
  lista.push_front(1);
  lista.push_back(8);

  cout << lista.pop_front() << "\n";

  if(lista.empty())
        cout << "Lista jest pusta\n";

  return 0;
}
```

Container	Insert Head	Insert Tail	Insert	Remove Head	Remove Tail	Remove	Index Search	Find
vector	n/a	O(1)	O(n)	O(1)	O(1)	O(n)	O(1)	O(log n)
list	O(1)	O(1)	O(1)	O(1)	O(1)	O(1)	n/a	O(n)
deque	O(1)	O(1)	n/a	O(1)	O(1)	O(n)	n/a	n/a
queue	n/a	O(1)	n/a	O(1)	n/a	n/a	O(1)	O(log n)
stack	O(1)	n/a	n/a	O(1)	n/a	n/a	n/a	n/a
map	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)	O(log n)
multimap	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)*	O(log n)
set	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)	O(log n)
multiset	n/a	n/a	O(log n)	n/a	n/a	O(log n)	O(1)*	O(log n)

http://devmentor.org/references/stl/stl.php

http://www.cplusplus.com/reference

Iteratory

```
for(list<int>::iterator it = lista.begin();
                        it != lista.end(); it++)
      cout << *it << endl;
```

Lepsza wersja:

```
for(list<int>::const_iterator it = lista.cbegin();
                        it != lista.cend(); it++)
      cout << *it << endl;
```

Iterator stały do mapy przechowującej pary liczb całkowitych indeksowane liczbami rzeczywistymi:

```
map<double, pair<int,int> >::const_iterator
```

Lepiej wykorzystać typ **`auto`** *!*