STL

Тема 3. Контейнеры.

Часть 3. Ассоциативные контейнеры. Set и Multiset

Set

- Отсортированный ассоциативный контейнер
- Хранящиеся элементы одновременно являются ключами

```
template < class T, class Compare = less<T>,
                   class Alloc = allocator<T>>
```

Set. Typedef

Typedef	Описание
key_type	*Тип ключа*
key_compare	*Тип функции сравнения ключей*
value_type	*Тим хранимых данных*
value_compare	*Тип функция сравнения данных*
const_iterator	Константный итератор
const_pointer	*const Type*.* В общем случае определен аллокатором
const_reference	*const Type&.* В общем случае определен аллокатором
const_reverse_iterator	Константный обратный итератор
difference_type	Знаковый целочисленный тип, который может определеть разность между двумя итераторами
iterator	итератор
pointer	*Type*.* В общем случае определен аллокатором
reference	*Type&.* В общем случае определен аллокатором
reverse_iterator	Обратный итератор
size_type	Тип представляющий количество элеменотов

Set. Constructor

```
set()
```

```
explicit set (const key_compare& comp = key_compare(),
              const allocator_type& alloc = allocator_type());
```

```
set (const set& x);
```

```
template <class InputIterator>
set (InputIterator first, InputIterator last,
    const key_compare& comp = key_compare(),
    const allocator_type& = allocator_type());
```

Set. Constructor

```
bool fncomp (int lhs, int rhs)  { return lhs<rhs; }


int main ()
{
   std::set<int> first;

    int myints[]= {10,20,30,40,50};
   std::set<int> second (myints,myints+5);
   std::set<int> third (second
   std::set<int> fourth (second.begin(), second.end());
   std::set<int,classcomp> fifth;
}
```

Set::Insert

```
pair<iterator,bool> insert (const value_type& val);

iterator insert (const_iterator position, const value_type& val);

template <class InputIterator>
void insert (InputIterator first, InputIterator last);
```

Set::erase

iterator erase(const_iterator position);

size_type erase(const value_type& val);

iterator erase(const_iterator first, const_iterator last);

Set::clear и Set::swap

```
void clear();
```

```
void swap (set& x);
```

set::find

iterator find (const value_type& val);

```
std::set<int> myset; std::set<int>::iterator it;

for (int i=1; i<=5; i++)
        myset.insert(i*10);

it=myset.find(20);

myset.erase (it);

myset.erase (myset.find(40));
```

Set::count

size_type count (const value_type& val) const;

```
std::set<int> myset;
for (int i=1; i<5; ++i)
            myset.insert(i*3);

for (int i=0; i<10; ++i)
{
    std::cout << i;
    if (myset.count(i)!=0) std::cout << " is an element of myset.\n“;
}
```

Аксессоры

- iterator begin();
- iterator end();
- reverse_iterator rbegin();
- reverse_iterator rend();
- bool empty();
- size_type size();
- size_type max_size();

Set::Lower_bound, upper_bound

iterator lower_bound (const value_type& val);

iterator upper_bound (const value_type& val);

pair<iterator,iterator> equal_range (const value_type& val);

```
std::set<int> myset;
std::set<int>::iterator itlow,itup;
for (int i=1; i<10; i++)
            myset.insert(i*10);              // 10 20 30 40 50 60 70 80 90

itlow=myset.lower_bound (30);
itup=myset.upper_bound (60);

myset.erase(itlow,itup);                     // 10 20 70 80 90
```

Практическое задание

1. Даны два текста. Найти самое короткое слово первого текста, которого нет во втором (с использованием множеств).