STL

Тема 3. Контейнеры.

Часть 4. Ассоциативные контейнеры. Map

Map

- Отсортированный ассоциативный контейнер по ключу
- Уникальность ключа

```
template <typename Key, typename T,
          typename Compare = less<Key>,
          class Allocator = allocator<pair<const Key, T> > >
```

Map. Typedef

Typedef	Описание
key_type	*Тип ключа*
key_compare	*Тип функции сравнения ключей*
mapped_type	*Тим хранимых данных*
value_type	**typedef pair<const Key, T> value_type;**
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

Map. Constructor

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

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

```
map (const map& x);
```

Map. Constructor

```
int main () {
    std::map<char,int> first;
    first['a']=10;
    first['b']=30;
    first['c']=50;
    first['d']=70;

    std::map<char,int> second (first.begin(),first.end());

    std::map<char,int> third (second);
}
```

Map::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);
```

Map::operator[]

mapped_type& operator[] (const key_type& k);

```
std::map<char,std::string> mymap;
mymap['a']="an element";
mymap['b']="another element";
Mymap[‘a’]=“renew element”
```

Map::erase

```
void erase (iterator position);

size_type erase (const key_type& k);

void erase (iterator first, iterator last);
```

Map::clear и Map::swap

```
void clear();
```

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

Map::find

iterator find (const key_type& val);

```
std::map<char,int> mymap;
std::map<char,int>::iterator it;
mymap['a']=50;
mymap['b']=100;
mymap['c']=150;
mymap['d']=200;

it=mymap.find('b');
mymap.erase (it);
mymap.erase (mymap.find('d'));
```

Map::count

size_type count (const key_type& k val) const;

```
std::map<char,int> mymap;
char c;
mymap ['a']=101;
mymap ['c']=202;
mymap ['f']=303;

for (c='a'; c<'h'; c++) {
        std::cout << c;
        if (mymap.count(c)>0)  std::cout << " is an element of mymap.\n";
        else std::cout << " is not an element of mymap.\n";
}
```

Map::Lower_bound, upper_bound

iterator lower_bound (const key_type& val);

iterator upper_bound (const key_type& val);

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

```
std::map<char,int> mymap;
std::map<char,int>::iterator itlow,itup;
mymap['a']=20;
mymap['b']=40;
mymap['c']=60;
mymap['d']=80; mymap['e']=100;

itlow=mymap.lower_bound ('b');
itup=mymap.upper_bound ('d');

mymap.erase(itlow,itup); // erases [itlow,itup)
```

Аксессоры

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

Практическое задание

1. Дан текст.
    1. Подсчитать количество разлинчых слов в данном тексте.
    2. Найти слово встречающееся максимальное количество раз
2. Переписать задачу с оценками учащихся с использованием map