Стандартная библиотека шаблонов

Ассоциативные контейнеры

Свойства ассоциативных контейнеров

Ассоциативные контейнеры обеспечивают быстрый поиск данных, основанных на ключах.

Библиотека STL предоставляет четыре основных вида контейнеров:

set (множество),

multiset (множество с дубликатами),

map (словарь или карта) и

multimap (словарь с дубликатами).

Являются двунаправленными контейнерами

Свойства ассоциативных контейнеров

Все ассоциативные контейнеры имеют в качестве параметров *Key* (ключ) и упорядочивающее отношение *Compare*, которое вызывает полное упорядочение по элементам *Key*.

Кроме того, *map* и *multimap* ассоциируют произвольный тип *T* с *Key,* образуя пару *pair<const Key, T>*

Объект типа *Compare* называется *сравнивающим объектом* (*comparison object*) контейнера.

Свойства ассоциативных контейнеров

Ассоциативный контейнер поддерживает *уникальные ключи* (*unique keys*), если он может содержать, самое большее, один элемент для каждого значения ключа.

Иначе он поддерживает *равные ключи* (*equal keys*).

set и *map* поддерживают уникальные ключи.

multiset и *multimap* поддерживают равные ключи.

Свойства ассоциативных контейнеров

Итераторы ассоциативного контейнера относятся к двунаправленным итераторам.

Операция *insert* не влияет на действительность итераторов и ссылок контейнера.

Операция *erase* делает недействительными только итераторы и ссылки на удаленные элементы.

Класс set

- Ассоциативный, обратимый, отсортированный контейнер с уникальными ключами, которые совпадают с хранимыми значениями
- По умолчанию, при сортировке используется сравнение «меньше» (less)

Основные методы

insert(key)

erase(key)

clear()

iterator find(key)

iterator lower_bond(key)

iterator upper_bound(key)

int count(key)

int size()

iterator begin()

iterator end()

Алгоритмы для множеств

- set_union(. . .)
- set_intersection(. . .)
- set_difference(. . .)

Пример

```
set <char> s,s1,s2;
s.insert('a');s.insert('s'); s.insert('b');
s1.insert('y'); s1.insert('a'); s1.insert('s');
ostream_iterator <char> co(cout," ");
set_intersection(s.begin(),     s.end(),
                 s1.begin(),s1.end(),
                 co);
cout<<endl;
```

Пример

```
insert_iterator <set<char> > it (s2, s2.begin());
set_union(s.begin(),s.end(),
                  s1.begin(),s1.end(),
                  it);
set <char>::iterator i;
for(i=s2.begin();i!=s2.end();i++)
        cout<<*i<<"  ";
```

multiset

- Поддерживает множественные копии ключа (возможно реализуемые в виде счетчика)
- операция erase(key) удаляет все вхождения ключа, возвращая количество удаленных
- операция count (key) считает количество равных key

map

- ассоциативный контейнер, который поддерживает уникальные ключи (не содержит ключи с одинаковыми значениями) и обеспечивает быстрый поиск значений другого типа *T*, связанных с ключами
- в качестве хранящихся значений используются пары «ключ-значение» (шаблонный класс pair <Key,T>)
- для создания пары можно использовать функцию

  make_pair (valKey, valT)

map

- Добавление insert
- Удаление erase
- Поиск по ключу find, lower_bound, upper_bound
- Доступ к элементам карты возможен через перегруженную операцию индексирования
- clear, count, size, empty

Пример

```
map<char, int >  m;
pair<const char, int>  item ('c', 100);

m.insert(item);
m.insert(pair<const char, int> ('x', 20);

m[ 'v‘ ]=5;
```

Пример

```
map<char,int>::iterator it

for (it=m.begin();it!=m.end(); it++)
    cout<<(*it).first<<"   "<<(*it).second<<endl;
```

Итератор - двунаправленный

Пример

```
it=m.find('a');
if (it!=m.end())
  cout<<(*it).first<<"   "<<(*it).second<<endl;
  else cout<<"NOT"<<endl;
```

multimap

- В отличие от map не имеет перегруженной операции индексирования
- Функция count для заданного ключа считает количество элементов с таким ключем

Пример

```
#include <iostream>
#include <map>
using namespace std;

typedef multimap<char,int> AbcMap;
typedef pair<char,int> Pair;
```

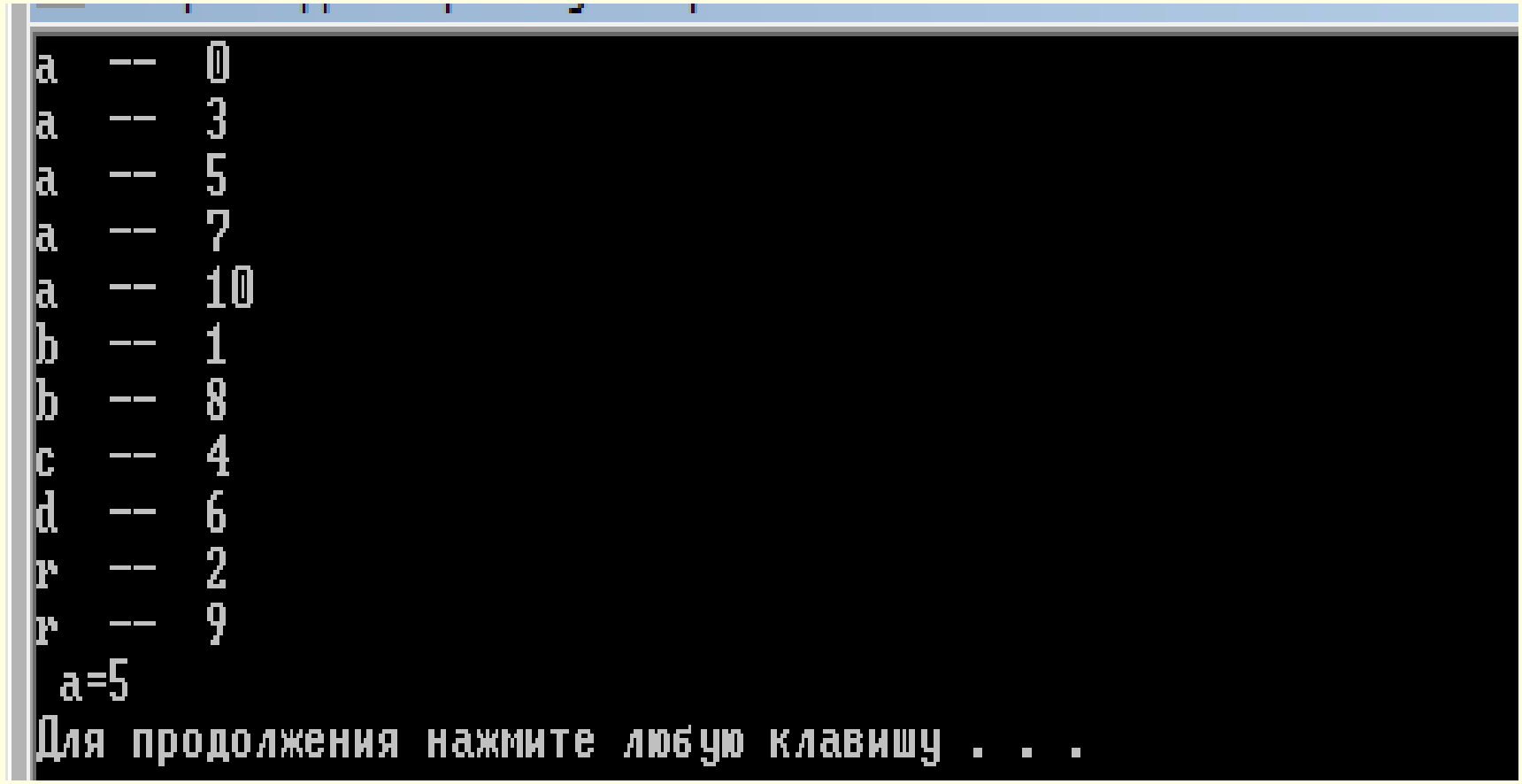
Пример

```
// Создание
    AbcMap m;
    char str[80]="abracadabra";
    char *pc=str;
    int i=0;
    while (*pc){
          m.insert(make_pair(*pc,i));
          pc++;
          i++;
    }
```

Пример

```
AbcMap::iterator it;
for(it=m.begin(); it!=m.end(); it++)
   cout<<(*it).first<<"  --  "<<(*it).second<<endl;

   cout<<" a="<<m.count('a')<<endl;
```


a -- 0
a -- 3
a -- 5
a -- 7
a -- 10
b -- 1
b -- 8
c -- 4
d -- 6
r -- 2
r -- 9
a=5
Для продолжения нажмите любую клавишу . . .

multimap

- Имеется функция, которая по заданному ключу возвращает пару итераторов, указывающих на диапазон элементов с таким ключом

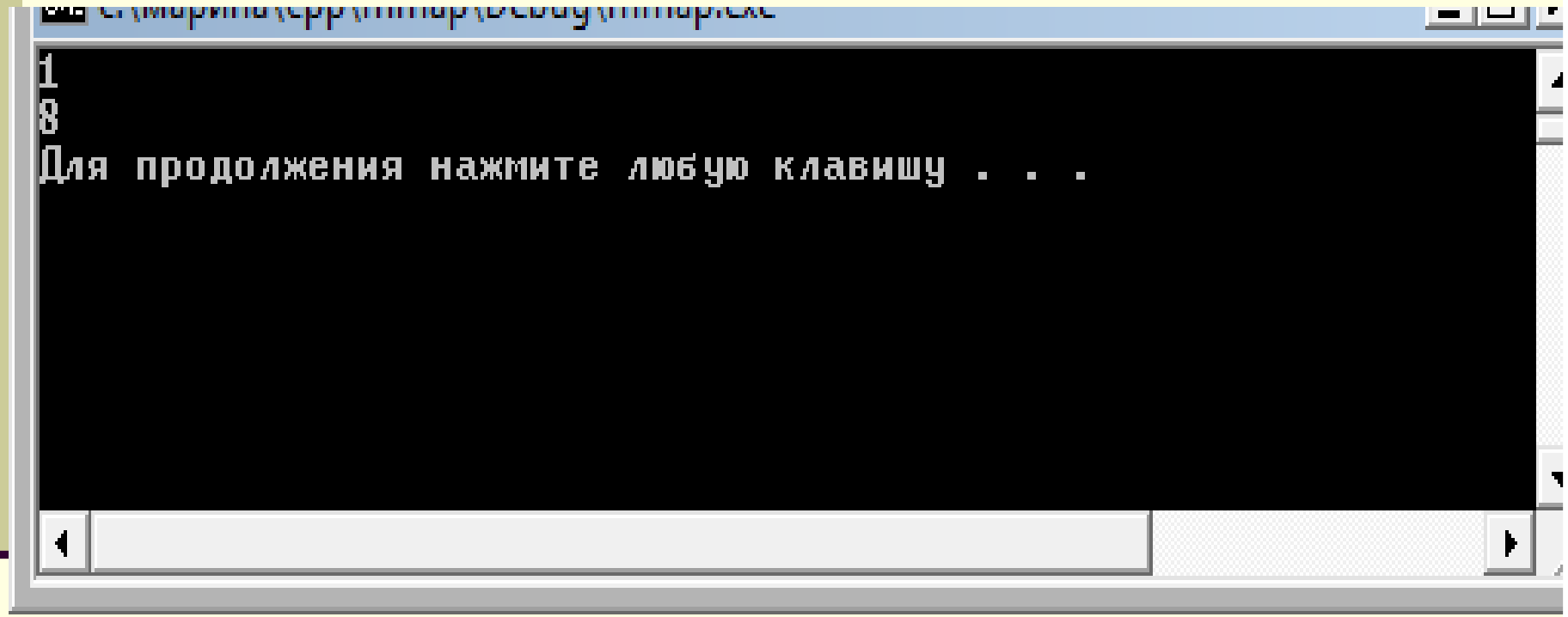
  equal_range

Пример

```
pair<AbcMap::iterator, AbcMap::iterator> range;

range = m.equal_range('b');

for(it=range.first; it!=range.second; it++)
        cout<<(*it).second<<endl;
```


1
8
Для продолжения нажмите любую клавишу . . .