无序容器：hash table。

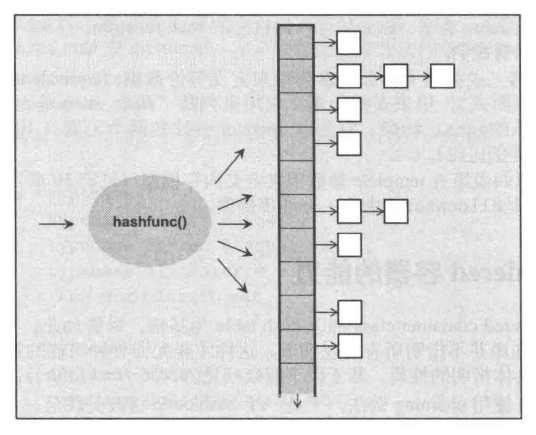
无序容器以一种随意顺序包含你插入的所有元素。和set，map相比起来，没有排序准则，和sequence容器比起来，没有办法放某个元素到特定位置。

unordered\_set和unordered\_multiset同set与multiset类似，元素必须具备comparable，尽管其没有排序准则。

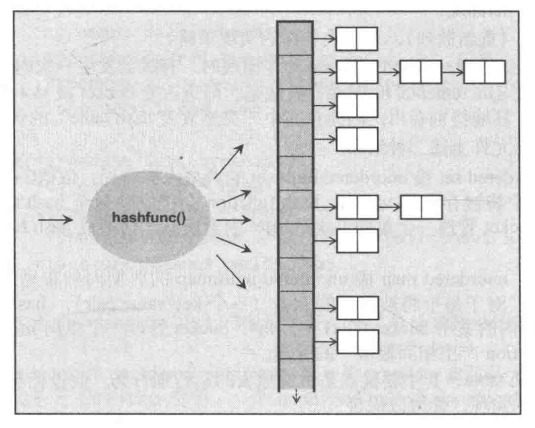
unordered\_map和unordered\_multimap同map与multimap也类似，key/value必须具备copyable或moveable，key必须可comparable。

STL中，unordered\_set和unordered\_map的冲突解决策略貌似是链表法。

unordered\_set和unordered\_multiset



unordered\_map和unordered\_multimap



例：程序stl\_test51

class Customer {

public:

Customer(const *string*& fn, const *string*& ln, long n)

: fname\_(fn), lname\_(ln), no\_(n)

{

}

friend *ostream*& operator << (*ostream*& os, const Customer& c)

{

return os << "[" << c.fname\_ << "," << c.lname\_

<< "," << c.no\_ << "]";

}

friend class CustomerHash;

friend class CustomerEqual;

private:

*string* fname\_;

*string* lname\_;

long no\_;

};

class CustomerHash

{

public:

*size\_t* operator() (const Customer& c) const {

return hash\_val(c.fname\_, c.lname\_, c.no\_);

}

};

class CustomerEqual

{

public:

bool operator() (const Customer& c1, const Customer& c2) const

{

return c1.no\_ == c2.no\_;

}

};

int *main*(int argc, char\* argv[])

{

// CustomerHash为自己提供的hash函数

// CustomerEqual为自己提供的等价准则

*unordered\_set*<Customer, CustomerHash, CustomerEqual> custset;

custset.*insert*(Customer("nico", "josuttis", 42));

PrintElements(custset);

return 0;

}

输出为：

[nico,josuttis,42]

查看hash表内部的状态

例：程序stl\_test52

template <typename T, typename V>

*ostream*& operator << (*ostream*& os, *pair*<T, V>& p)

{

return os << "[" << p.*first* << "," << p.*second* << "]";

}

template<typename T>

inline void PrintHashTableState(const T& cont)

{

*cout* << "size: " << cont.*size*() << *endl*;

*cout* << "buckets: " << cont.*bucket\_count*() << *endl*;

*cout* << "load factor: " << cont.*load\_factor*() << *endl*;

*cout* << "max load factor " << cont.*max\_load\_factor*() << *endl*;

if (typeid(typename *iterator\_traits*<typename T::*iterator*>::*iterator\_category*)

== typeid(*bidirectional\_iterator\_tag*))

{

*cout* << "chaining style: doubly-linked" << *endl*;

}

else {

*cout* << "chaining style: singly-linked" << *endl*;

}

*cout* << "data: " << *endl*;

for (auto idx = 0; idx != cont.*bucket\_count*(); ++idx)

{

*cout* << "b[" << *setw*(2) << idx << "]: ";

for (auto pos = cont.*begin*(idx); pos != cont.*end*(idx); ++pos)

{

*cout* << \*pos << " ";

}

*cout* << *endl*;

}

*cout* << *endl*;

}

*unordered\_set*<int> intset = {1, 2, 3, 5, 7, 11, 13, 17, 19};

PrintHashTableState(intset);

*cout* << "insert some numbers" << *endl*;

intset.*insert*({ -7, 17, 33, 4 });

PrintHashTableState(intset);

输出为：

size: 9

buckets: 64

load factor: 0.140625

max load factor 1

chaining style: doubly-linked

data:

b[ 0]: 5

b[ 1]:

b[ 2]:

b[ 3]:

b[ 4]: 1

b[ 5]:

b[ 6]:

b[ 7]:

b[ 8]: 13

b[ 9]:

b[10]:

b[11]:

b[12]:

b[13]:

b[14]:

b[15]:

b[16]:

b[17]:

b[18]:

b[19]:

b[20]: 17

b[21]:

b[22]:

b[23]:

b[24]:

b[25]:

b[26]:

b[27]:

b[28]:

b[29]:

b[30]:

b[31]:

b[32]:

b[33]:

b[34]: 7

b[35]:

b[36]:

b[37]:

b[38]: 3

b[39]:

b[40]:

b[41]:

b[42]:

b[43]:

b[44]:

b[45]:

b[46]: 11

b[47]:

b[48]:

b[49]:

b[50]:

b[51]:

b[52]:

b[53]:

b[54]: 19

b[55]: 2

b[56]:

b[57]:

b[58]:

b[59]:

b[60]:

b[61]:

b[62]:

b[63]:

insert some numbers

size: 12

buckets: 64

load factor: 0.1875

max load factor 1

chaining style: doubly-linked

data:

b[ 0]: 5

b[ 1]:

b[ 2]:

b[ 3]:

b[ 4]: 1

b[ 5]:

b[ 6]:

b[ 7]:

b[ 8]: 13

b[ 9]:

b[10]:

b[11]:

b[12]:

b[13]:

b[14]:

b[15]: -7

b[16]:

b[17]: 4

b[18]:

b[19]:

b[20]: 17

b[21]:

b[22]:

b[23]:

b[24]:

b[25]:

b[26]:

b[27]:

b[28]:

b[29]:

b[30]:

b[31]:

b[32]:

b[33]:

b[34]: 7

b[35]:

b[36]: 33

b[37]:

b[38]: 3

b[39]:

b[40]:

b[41]:

b[42]:

b[43]:

b[44]:

b[45]:

b[46]: 11

b[47]:

b[48]:

b[49]:

b[50]:

b[51]:

b[52]:

b[53]:

b[54]: 19

b[55]: 2

b[56]:

b[57]:

b[58]:

b[59]:

b[60]:

b[61]:

b[62]:

b[63]:

另一个hash map

例：程序stl\_test52

*unordered\_multimap*<*string*, *string*> dict = {

{"day", "Tag"},

{"strange", "fremd"},

{"car", "Auto"},

{"smart", "elegant"},

{"trait", "Merkmal"},

{"strange", "seltsam"}

};

PrintHashTableState(dict);

*cout* << "insert some pair" << *endl*;

dict.*insert*({ {"smart", "raffiniert"}, {"smart", "klug"}, {"clever", "raffiniert"} });

PrintHashTableState(dict);

*cout* << "set max load factor" << *endl*;

dict.*max\_load\_factor*(0.7f);

PrintHashTableState(dict);

输出为：

size: 6

buckets: 8

load factor: 0.75

max load factor 1

chaining style: doubly-linked

data:

b[ 0]:

b[ 1]: [car,Auto]

b[ 2]: [smart,elegant]

b[ 3]: [trait,Merkmal]

b[ 4]:

b[ 5]: [day,Tag]

b[ 6]:

b[ 7]: [strange,fremd] [strange,seltsam]

insert some pair

size: 9

buckets: 64

load factor: 0.140625

max load factor 1

chaining style: doubly-linked

data:

b[ 0]:

b[ 1]:

b[ 2]:

b[ 3]:

b[ 4]:

b[ 5]:

b[ 6]:

b[ 7]:

b[ 8]:

b[ 9]:

b[10]:

b[11]:

b[12]:

b[13]:

b[14]:

b[15]: [strange,fremd] [strange,seltsam]

b[16]:

b[17]:

b[18]: [smart,elegant] [smart,raffiniert] [smart,klug]

b[19]:

b[20]:

b[21]:

b[22]:

b[23]:

b[24]:

b[25]:

b[26]:

b[27]:

b[28]:

b[29]:

b[30]:

b[31]:

b[32]:

b[33]: [car,Auto]

b[34]:

b[35]:

b[36]:

b[37]:

b[38]:

b[39]:

b[40]:

b[41]:

b[42]:

b[43]: [trait,Merkmal]

b[44]:

b[45]:

b[46]:

b[47]:

b[48]:

b[49]:

b[50]:

b[51]:

b[52]:

b[53]:

b[54]:

b[55]:

b[56]:

b[57]:

b[58]: [clever,raffiniert]

b[59]:

b[60]:

b[61]: [day,Tag]

b[62]:

b[63]:

set max load factor

size: 9

buckets: 64

load factor: 0.140625

max load factor 0.7

chaining style: doubly-linked

data:

b[ 0]:

b[ 1]:

b[ 2]:

b[ 3]:

b[ 4]:

b[ 5]:

b[ 6]:

b[ 7]:

b[ 8]:

b[ 9]:

b[10]:

b[11]:

b[12]:

b[13]:

b[14]:

b[15]: [strange,fremd] [strange,seltsam]

b[16]:

b[17]:

b[18]: [smart,elegant] [smart,raffiniert] [smart,klug]

b[19]:

b[20]:

b[21]:

b[22]:

b[23]:

b[24]:

b[25]:

b[26]:

b[27]:

b[28]:

b[29]:

b[30]:

b[31]:

b[32]:

b[33]: [car,Auto]

b[34]:

b[35]:

b[36]:

b[37]:

b[38]:

b[39]:

b[40]:

b[41]:

b[42]:

b[43]: [trait,Merkmal]

b[44]:

b[45]:

b[46]:

b[47]:

b[48]:

b[49]:

b[50]:

b[51]:

b[52]:

b[53]:

b[54]:

b[55]:

b[56]:

b[57]:

b[58]: [clever,raffiniert]

b[59]:

b[60]:

b[61]: [day,Tag]

b[62]:

b[63]:

从输出可以看出，相同的key值hash到了同一个bucket，形成了链表。