

#include <iostream>

using namespace std;

#include <string>

#include<vector>

#include<algorithm>

void test01()

{

vector<int> v;

for (int i = 0; i < 10; i++)

{

v.push\_back(i);

}

vector<int> ::iterator it= find(v.begin(), v.end(), 50);

if (it == v.end())

{

cout << "没有找到。" << endl;

}

else

{

cout << "找到了： " << \*it << endl;

}

}

class Person

{

public:

Person(string name, int age)

{

this->m\_Age = age;

this->m\_Name = name;

}

//重载 == 底层find 不知道如何比较Person中的两个数据类型。

bool operator == (const Person &p)

{

if (this->m\_Name == p.m\_Name &&this->m\_Age == p.m\_Age)

{

return true;

}

else

{

return false;

}

}

string m\_Name;

int m\_Age;

};

void test02()

{

vector<Person> v;

Person p1("aaa", 10);

Person p2("bbb", 20);

Person p3("ccc", 30);

Person p4("ddd", 40);

Person p5("fff", 50);

v.push\_back(p1);

v.push\_back(p2);

v.push\_back(p3);

v.push\_back(p4);

v.push\_back(p5);

vector<Person>::iterator it = find(v.begin(), v.end(), p2);

if (it == v.end())

{

cout << "没有找到。" << endl;

}

else

{

cout << "找到了：姓名 " << (\*it).m\_Name<<"年龄："<<(\*it).m\_Age<< endl;

}

}

int main()

{

test02();

system("pause");

return 0;

}



#include <iostream>

using namespace std;

#include <string>

#include<vector>

#include<algorithm>

class GreaterFive

{

public:

bool operator()(int val)

{

return val > 5;

}

};

void test01()

{

vector<int> v;

for (int i = 0; i < 10; i++)

{

v.push\_back(i);

}

vector<int>::iterator it = find\_if(v.begin(), v.end(), GreaterFive());

if (it == v.end())

{

cout << "没找到。" << endl;

}

else

{

cout << "找到大于5的数: " << \*it << endl;

}

}

class Person

{

public:

Person(string name, int age)

{

this->m\_Age = age;

this->m\_Name = name;

}

string m\_Name;

int m\_Age;

};

class Greater20

{

public:

bool operator()(Person &p)

{

return p.m\_Age > 20;

}

};

void test02()

{

vector<Person> v;

Person p1("aaa", 10);

Person p2("bbb", 20);

Person p3("ccc", 30);

Person p4("ddd", 40);

Person p5("fff", 50);

v.push\_back(p1);

v.push\_back(p2);

v.push\_back(p3);

v.push\_back(p4);

v.push\_back(p5);

vector<Person>::iterator it = find\_if(v.begin(), v.end(), Greater20());

if (it == v.end())

{

cout << "没找到。" << endl;

}

else

{

cout << "找到大于20的年龄的姓名： " << it->m\_Name << endl;

}

}

int main()

{

test02();

system("pause");

return 0;

}



#include <iostream>

using namespace std;

#include <string>

#include <vector>

#include <algorithm>

void test01()

{

vector<int> v;

v.push\_back(2);

v.push\_back(0);

v.push\_back(3);

v.push\_back(3);

v.push\_back(1);

v.push\_back(9);

vector<int> ::iterator pos = adjacent\_find(v.begin(), v.end());

if (pos == v.end())

{

cout << "未找到相邻重复元素。" << endl;

}

else

{

cout << "找到相邻重复元素。" << \*pos << endl;

}

}

int main()

{

test01();

system("pause");

return 0;

}



#include <iostream>

using namespace std;

#include <string>

#include <vector>

#include <algorithm>

void test01()

{

vector<int> v;

for (int i = 0; i < 10; i++)

{

v.push\_back(i);

}

//容器下必须是有序的序列。如果无序，结果未知。

bool ret = binary\_search(v.begin(), v.end(),5);

if (ret)

{

cout << "找到相邻重复元素。" << endl;

}

else

{

cout << "未找到相邻重复元素。" << endl;

}

}

int main()

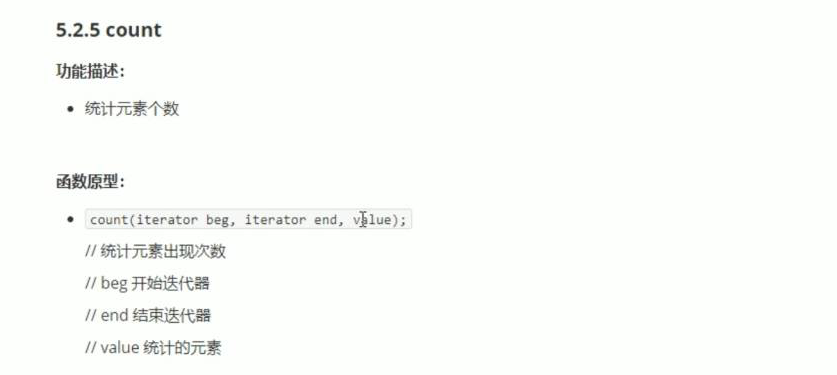
{

test01();

system("pause");

return 0;

}





#include <iostream>

using namespace std;

#include <string>

#include <vector>

#include <algorithm>

void test01()

{

vector<int> v;

v.push\_back(10);

v.push\_back(50);

v.push\_back(30);

v.push\_back(40);

v.push\_back(50);

int num = count(v.begin(), v.end(), 50);

cout << "50的个数为： " << num << endl;

}

class Person

{

public:

Person(string name, int age)

{

this->m\_Age = age;

this->m\_Name = name;

}

bool operator == ( const Person &p)

{

if (this->m\_Age == p.m\_Age)

{

return true;

}

else

{

return false;

}

}

string m\_Name;

int m\_Age;

};

void test02()

{

vector<Person> v;

Person p1("刘备", 30);

Person p2("关羽", 20);

Person p3("张飞", 20);

Person p4("赵云", 25);

Person p5("黄忠", 60);

v.push\_back(p1);

v.push\_back(p2);

v.push\_back(p3);

v.push\_back(p4);

v.push\_back(p5);

Person p("诸葛亮", 20);

int num = count(v.begin(), v.end(),p);

cout << "和诸葛亮一样岁数的人： " << num << endl;

}

int main()

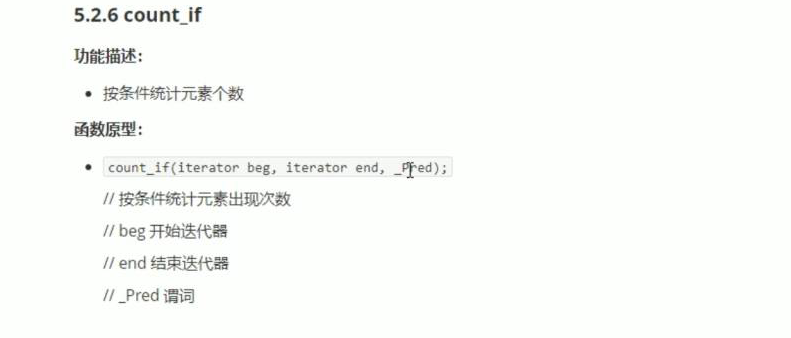
{

test02();

system("pause");

return 0;

}



#include <iostream>

using namespace std;

#include <string>

#include <vector>

#include <algorithm>

class Greater20

{

public:

bool operator()(int val)

{

return val > 20;

}

};

void test01()

{

vector<int> v;

v.push\_back(10);

v.push\_back(50);

v.push\_back(30);

v.push\_back(40);

v.push\_back(50);

int num = count\_if(v.begin(), v.end(), Greater20());

cout << "大于20的个数为： " << num << endl;

}

class Person

{

public:

Person(string name, int age)

{

this->m\_Age = age;

this->m\_Name = name;

}

bool operator == ( const Person &p)

{

if (this->m\_Age == p.m\_Age)

{

return true;

}

else

{

return false;

}

}

string m\_Name;

int m\_Age;

};

class AgeGreater20

{

public:

bool operator()(const Person&p)

{

return p.m\_Age > 20;

}

};

void test02()

{

vector<Person> v;

Person p1("刘备", 30);

Person p2("关羽", 20);

Person p3("张飞", 20);

Person p4("赵云", 25);

Person p5("黄忠", 60);

v.push\_back(p1);

v.push\_back(p2);

v.push\_back(p3);

v.push\_back(p4);

v.push\_back(p5);

int num = count\_if(v.begin(), v.end(), AgeGreater20());

cout << "年龄大于25的人数目： " << num << endl;

}

int main()

{

test02();

system("pause");

return 0;

}