//模板的局限性

#include <iostream>

#include<string>

using namespace std;

//模板不是万能的，有些特点的数据类型要具体化方式做特殊实现

class person {

public:

person(string name, int age) {

this->mname = name;

this->mage = age;

}

string mname;

int mage;

};

//对比两个数据是否相等函数

template<class T>

bool myCompare(T& a, T& b) {

if (a == b) {

return true;

}

else {

return false;

}

}

//利用具体化person的版本，来实现代码，具体化优先调用

template<> bool myCompare(person& p1, person& p2) {

if (p1.mname == p2.mname && p1.mage == p2.mage) {

return true;

}

else {

return false;

}

}

void test01() {

int a = 10;

int b = 20;

bool ret = myCompare(a, b);

if (ret) {

cout << "a==b" << endl;

}

else {

cout << "a!=b" << endl;

}

}

void test02() {

person p1("ton", 10);

person p2("ton", 10);

bool ret = myCompare(p1, p2);

if (ret) {

cout << "p1==p2" << endl;

}

else {

cout << "p1!=p2" << endl;

}

}

int main() {

test01();

test02();

system("pause");

return 0;

}