函数模板：不是真正的函数，而是产生函数的公式。

// Copyright 2015.刘珅珅

// author：刘珅珅

// 函数模板的测试

#include <iostream>

#include <vector>

using namespace std;

template<class T>

const T& FindMax(const vector<T>& array)

{

int max\_index = 0;

for (int i = 0; i != array.size(); ++i)

{

if (array[max\_index] < array[i])

{

max\_index = i;

}

}

return array[max\_index];

}

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

{

vector<int> array;

array.push\_back(1);

array.push\_back(2);

array.push\_back(3);

cout << FindMax(array) << endl;

return 0;

}

注意：在函数重载时，如果有一个非模板和一个模板可以匹配，非模板具有优先匹配权。

类模板：类模板及其实现放在一个单独的头文件中，遵循C++ STL库的方式，不再分离成头文件和源文件。

函数对象：定义类包含成员函数和0个数据成员，其对象为函数对象。

// Copyright 2015.刘珅珅

// author：刘珅珅

// 函数模板的测试：使用函数对象

#include <functional>

#include <iostream>

#include <string>

#include <vector>

using namespace std;

template <typename Object, typename Comparator>

const Object& FindMax(const vector<Object>& array, Comparator isLessThan)

{

int max\_index = 0;

for (int i = 0; i != array.size(); ++i)

{

if (isLessThan(array[max\_index], array[i]))

{

max\_index = i;

}

}

return array[max\_index];

}

template <typename Object>

const Object& FindMax(const vector<Object>& array)

{

return FindMax(array, less<Object>());

}

class CaseInsensitiveCompare

{

public:

//重载函数调用操作符()

bool operator()(const string& lhs, const string& rhs) const

{

return \_stricmp(lhs.c\_str(), rhs.c\_str()) < 0;

}

};

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

{

vector<string> array(3);

array[0] = "ZEBRA";

array[1] = "alligator";

array[2] = "crocodile";

cout << FindMax(array, CaseInsensitiveCompare()) << endl;

cout << FindMax(array) << endl;

return 0;

}