



程序设计实习

郭炜 微博 <http://weibo.com/guoweiofpku>

<http://blog.sina.com.cn/u/3266490431>

刘家瑛 微博 <http://weibo.com/pkuliujiaying>



C++11特性

无序容器(哈希表)

```
#include <iostream>
#include <string>
#include <unordered_map>
using namespace std;
int main()
{
    unordered_map<string,int> turingWinner; //图灵奖获奖名单
    turingWinner.insert(make_pair("Dijkstra",1972));
    turingWinner.insert(make_pair("Scott",1976));
    turingWinner.insert(make_pair("Wilkes",1967));
    turingWinner.insert(make_pair("Hamming",1968));
    turingWinner["Ritchie"] = 1983;
    string name;
    cin >> name; //输入姓名
```

```
unordered_map<string,int>::iterator p = turingWinner.find(name);  
//据姓名查获获奖时间  
if( p != turingWinner.end())  
    cout << p->second;  
else  
    cout << "Not Found" << endl;  
return 0;  
}
```

哈希表插入和查询的时间复杂度几乎是常数

正则表达式

```
#include <iostream>
#include <regex> //使用正则表达式须包含此文件
using namespace std;
int main()
{
    regex reg("b.?p.*k");
    cout << regex_match("bopggk",reg) <<endl;    //输出 1, 表示匹配成功
    cout << regex_match("boopgggk",reg) <<endl;    //输出 0, 表示匹配失败
    cout << regex_match("b pk",reg) <<endl;    //输出 1, 表示匹配成功
    regex reg2("\\d{3}([a-zA-Z]+).\\d{2}|N/A)\\s\\1");
    string correct="123Hello N/A Hello";
    string incorrect="123Hello 12 hello";
    cout << regex_match(correct,reg2) <<endl;    //输出 1, 表示匹配成功
    cout << regex_match(incorrect,reg2) << endl;    //输出 0, 表示匹配失败
}
```

Lambda表达式

只使用一次的函数对象，能否不要专门为其编写一个类？

只调用一次的简单函数，能否在调用时才写出其函数体？

Lambda表达式

形式:

[外部变量访问方式说明符] (参数表) ->返回值类型

{

语句组

}

[=] 以传值的形式使用所有外部变量

[] 不使用任何外部变量

[&] 以引用形式使用所有外部变量

[x, &y] x 以传值形式使用, y 以引用形式使用

[=, &x, &y] x, y 以引用形式使用, 其余变量以传值形式使用

[&, x, y] x, y 以传值的形式使用, 其余变量以引用形式使用

“->返回值类型”也可以没有, 没有则编译器自动判断返回值类型。

Lambda表达式

```
int main()
{
    int x = 100,y=200,z=300;
    cout << [ ](double a,double b) { return a + b; } (1.2,2.5) << endl;
    auto ff = [=,&y,&z](int n) {
        cout <<x << endl;
        y++; z++;
        return n*n;
    };
    cout << ff(15) << endl;
    cout << y << "," << z << endl;
}
```


Lambda表达式

```
int main()
{
    int x = 100,y=200,z=300;
    cout << [ ](double a,double b) { return a + b; } (1.2,2.5) << endl;
    auto ff = [=,&y,&z](int n) {
        cout <<x << endl;
        y++; z++;
        return n*n;
    };
    cout << ff(15) << endl;
    cout << y << "," << z << endl;
}
```

输出：
3.7
100
225
201,301

Lambda表达式

```
int a[4] = { 4,2,11,33};  
sort(a,a+4,[ ](int x,int y)->bool { return x%10 < y%10; });  
for_each(a,a+4,[ ](int x) {cout << x << " " ;} ) ;
```

Lambda表达式

```
int a[4] = { 4,2,11,33};  
sort(a,a+4,[ ](int x,int y)->bool { return x%10 < y%10; });  
for_each(a,a+4,[ ](int x) {cout << x << " ";} ) ;
```

输出：
11 2 33 4

Lambda表达式

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main()
{
    vector<int> a { 1,2,3,4};
    int total = 0;
    for_each(a.begin(),a.end(),[&](int & x) {total += x; x*=2;});
    cout << total << endl; //输出 10
    for_each(a.begin(),a.end(),[](int x) { cout << x << " ";});
    return 0;
}
```

程序输出结果:

10

2 4 6 8

Lambda表达式

实现递归求斐波那契数列第n项：

```
function<int(int)> fib = [&fib](int n)
{ return n <= 2 ? 1 : fib(n-1) + fib(n-2);};
```

```
cout << fib(5) << endl; //输出5
```

`function<int(int)>` 表示返回值为 `int`，有一个 `int` 参数的函数

In-video Quiz

下面程序的输出结果是：

```
int n = 0;  
int a[] = {1,2,3,4 };  
for_each(a,a+4,[&](int e) { ++e; n += e; });  
cout << n << ", " << a[2] << endl;
```

A)14,3

B)14,4

C)10,3

D)10,4

In-video Quiz

下面程序的输出结果是：

```
int n = 0;  
int a[] = {1,2,3,4 };  
for_each(a,a+4,[&](int e) { ++e; n += e; });  
cout << n << ", " << a[2] << endl;
```

A)14,3

B)14,4

C)10,3

D)10,4

#A