Boost: Lambda

Before C++11, you needed to use a library like [Boost.Lambda](http://www.boost.org/libs/lambda" \t "_top) to take advantage of lambda functions. Since C++11, this library can be regarded as deprecated because lambda functions are now part of the programming language.

Lambda expression (anonymous function): Lambda is a function object. Lambda can be used instead of std :: function. Lamda in C ++ can capture local variables. The return value of lambda can be deduced type (meaning the keyword auto is used).

C ++ lambda has 4 syntax declarations as follows:

[ capture-list ] ( params ) mutable(optional) exception attribute -> ret { body } (1)

[ capture-list ] ( params ) -> ret { body } (2)

[ capture-list ] ( params ) { body } (3)

[ capture-list ] { body } (4)

1. Simple Lambda:

int **main**() {

[](){};

return 0;

}

2. The meaning of each parenthesis

int main () {

[] // Declare lambda

 () // Argument

 {} // Body function

  (); // Run!

  return 0; }

**3. Transmission of arguments**

**#include<iostream>**

int main() {

[](const std::string& x) { std::cout << x << std::endl; }("Hello lambda\n");

return 0;

}

4. Return value

#include <iostream>

int main () { auto x = [] (int a) -> int {return a + 5; } (10);

     std :: cout << x;

     return 0; }

The result is 15. Here -> int is the expressive spelling that this function will return an int. auto will deduce the type of x by itself.

5. Capture / Capture

Lambda features it as a Function Object and accesses a variable outside the scope (Scope created by its {}). Accessing a captured variable has two ways:

Access by reference.

Access by copying (by value).

In Lambda, to capture by reference, write [&]. To capture by value, write [=]. To not capture anything, leave [] blank.

6. . Capture with value will error when compile. If you still want to edit the capture value by value? -> Use mutable.

The variable a is passed by value and mutable so the value of the lambda is changed but when it comes out of the lambda scope, everything goes back to normal.

#include <iostream>

int main () {  int a = 5;

 [=] () mutable {a = a + 1; std :: cout << a << std :: endl; } (); // Print 6

 std :: cout << a << std :: endl; // Print 5

 return 0; }

7. Use Lambda and std :: function

#include <functional>

#include <iostream>

void print\_num(int i)

{ [std::cout](http://en.cppreference.com/w/cpp/io/cout) << i << '**\n**'; }

int main()

{

// store a free function

std::function<void(int)> f\_display = print\_num;

f\_display(-9);

// store a lambda

std::function<void()> f\_display\_42 = []() { print\_num(42); };

f\_display\_42();

}