

Slides

Development > Programming Languages > C++

The C++ 20 Masterclass : From Fundamentals to Advanced

Learn and Master Modern C++ From Beginning to Advanced in Plain English : C++11, C++14, C++17, C++20 and More!

4.7 ★★★★★

Created by [Daniel Gakwaya](#)

Section : Lambda Functions

Slide intentionally left empty

Lambda Functions : Introduction

A mechanism to set up anonymous functions (without names). Once we have them set up, we can either give them names and call them , or we can even get them to do things directly.

Slide intentionally left empty

Declaring and using lambda functions

A mechanism to set up anonymous functions (without names). Once we have them set up, we can either give them names and call them , or we can even get them to do things directly.

```

Lambda function signature :
..... [capture list] (parameters) ->return type{
..... // Function body
..... }

        [](){
std::cout << "Hello World!" << std::endl;
};

```


Lambda function signature

```
1 // Lambda function signature :
2 ..... [capture list] (parameters) ->return type{
3 ..... // Function body
4 ..... }
5
6 auto func = [](){
7     std::cout << "Hello World!" << std::endl;
8 };
```

Lambda function signature

```
1 // Lambda function signature :  
2 ..... [capture list] (parameters) ->return type{  
3 ..... // Function body  
4 ..... }  
5  
6 auto func = [](){  
7     std::cout << "Hello World!" << std::endl;  
8 };  
9 func();
```

```
1 // ...
2
3 /*
4  * Call lambda function directly after definition
5  */
6
7 [](){
8     std::cout << "Hello World!" << std::endl;
9 }();
10
```

```
1 //
2 /*
3  * Lambda function that takes parameters
4  * */
5
6 [](double a, double b){
7     std::cout << "a + b : " << (a + b) << std::endl;
8 }(12.1,5.7);
9
10
```

```
1 // ...
2
3 /*
4  * Lambda function that returns something
5  */
6
7 auto result = [](double a, double b){
8     return (a + b);
9 }(12.1, 5.7);
10 std::cout << "result : " << result << std::endl;
11
12 // ...
13
```

```
1 // ...
2
3 // ...
4 /*
5  * Print result directly
6  * */
7 // ...
8
9 std::cout << "result : " << [](double a, double b){
10     return (a + b);
11 }(12.1,5.7) << std::endl;
```

```
1 // ...
2
3 /*
4  * Specify return type explicitly
5  */
6
7 auto result = [](double a, double b)->double{
8     return (a + b);
9 }(12.1,5.7);
10 std::cout << "result : " << result << std::endl;
```

Slide intentionally left empty

Capture lists

Capture lists

```
Lambda function signature :  
[capture list] (parameters) ->return type{  
    // Function body  
}
```

```
1 //Capture lists
2 double a{10};
3 double b{20};
4
5 auto func = [a,b]() {
6     std::cout << "a + b : " << a + b << std::endl;
7 };
8 func();
```

Capturing by value

```
//C++20
//Capturing by value : what we have in the lambda function
// is a copy
int c{42};

auto func = [c]() {
    std::cout << "Inner value : " << c << std::endl;
};

for(size_t i{} ; i < 5 ; ++i){
    std::cout << "Outer value : " << c << std::endl;
    func();
    ++c;
}
```

Capturing by reference

```
1 //Capturing by reference : Working on the original outside value
2
3 int c{42};
4
5 auto func = [&c]() {
6     std::cout << "Inner value : " << c << std::endl;
7 };
8
9 for(size_t i{} ; i < 5 ; ++i){
10     std::cout << "Outer value : " << c << std::endl;
11     func();
12     ++c;
13 }
```

Slide intentionally left empty

Capture all in context

Capture lists

```
Lambda function signature :  
[capture list] (parameters) ->return type{  
    // Function body  
}
```


Capture all by value

```
1 //C++20
2 //Capturing everything by value
3
4 int c{42};
5
6 auto func = [=]() {
7     std::cout << "Inner value : " << c << std::endl;
8 };
9
10 for(size_t i{} ; i < 5 ; ++i){
11     std::cout << "Outer value : " << c << std::endl;
12     func();
13     ++c;
14 }
```

Capture all by reference

```
1 //Capturing everything by reference
2
3 int c{42};
4 double d{12.1};
5
6 auto func = [&]() {
7     std::cout << "Inner value c : " << c << std::endl;
8     std::cout << "Inner value d : " << d << std::endl;
9 };
10
11 for(size_t i{} ; i < 5 ; ++i) {
12     std::cout << "Outer value c : " << c << std::endl;
13     std::cout << "Outer value d : " << d << std::endl;
14     func();
15     ++c;
16     d+= 0.5;
17 }
```

Slide intentionally left empty

Lambda functions : Summary

A mechanism to set up anonymous functions (without names). Once we have them set up, we can either give them names and call them , or we can even get them to do things directly.

Capture lists

```
Lambda function signature :  
[capture list] (parameters) ->return type{  
    // Function body  
}
```

What you know now

```
1000. Lambda function signature
1001. Give lambda function a name and call it
1002. Call lambda function directly after definition
1003. Lambda function that takes parameters
1004. Lambda function that returns something'
1005. Print result directly
1006. Specify return type explicitly
1007. Capture lists
1008. capture by value
1009. capture by reference
1010. capture all by value
1011. capture all by reference
```

1012. ...

Slide intentionally left empty