Slides

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

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

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

Lambda function signature

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

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

Lambda function signature

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

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

```
/*
 * Call lambda function directly after definition
 * */

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

```cpp
/*
 * Lambda function that takes parameters
 * */

[](double a, double b){
    std::cout << "a + b : " << (a + b) <<  std::endl;
}(12.1,5.7);
```

```
/*
 * Lambda function that returns something
 * */

auto result = [](double a, double b){
    return (a + b);
}(12.1,5.7);
std::cout << "result : " << result << std::endl;
```

```
/*
 * Print result directly
 * */

std::cout << "result : " <<  [](double a, double b){
    return (a + b);
}(12.1,5.7)  << std::endl;
```

```
/*
 * Specify return type explicitly
 * */
auto result = [](double a, double b)->double{
    return (a + b);
}(12.1,5.7);
std::cout << "result : " << result << std::endl;
```

Slide intentionally left empty

Capture lists

Capture lists

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

```
//Capture lists
double a{10};
double b{20};

auto func = [a,b](){
    std::cout  << "a + b : " << a + b << std::endl;
};
func();
```

Capturing by value

```
//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

```
//Capturing by reference : Working on the original outside value

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;
}
```

Slide intentionally left empty

Capture all in context

Capture lists

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

Capture all by value

```
//Capturing everything by value

int c{42};

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

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

Capture all by reference

```
//Capturing everything by reference

int c{42};
double d{12.1};

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

for(size_t i{} ; i < 5 ;++i){
    std::cout << "Outer value c : " << c << std::endl;
    std::cout << "Outer value d : " << d << std::endl;
    func();
    ++c;
    d+= 0.5;
}
```

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

- Lambda function signature
- Give lambda function a name and call it
- Call lambda function directly after definition
- Lambda function that takes parameters
- Lambda function that returns something'
- Print result directly
- Specify return type explicitly
- Capture lists
- capture by value
- capture by reference
- capture all by value
- capture all by reference

Slide intentionally left empty