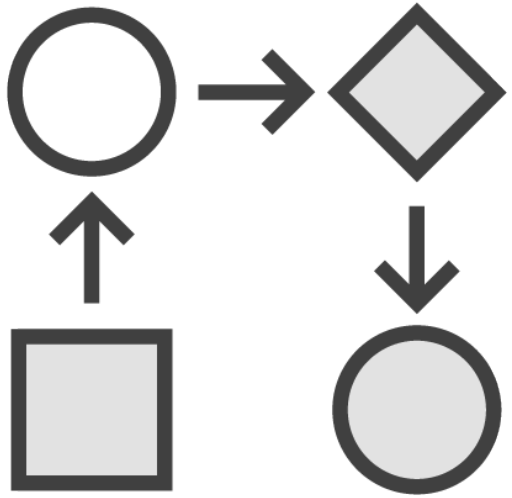
Lambdas

Kate Gregory

@gregcons www.gregcons.com/kateblog

What's a Lambda?

An expression that represents doing something



Imagine handing an operation or function (code) to some other operation or function

- For generic work
- For a functional style
- For concurrency
- For readability
  - Eliminate tiny functions

Tiny Functions

```
auto isOdd = [](int candidate) {return candidate % 2 != 0; };


bool is3Odd = isOdd(3);

bool is4Odd = isOdd(4);


vector nums { 2,3,4,-1,1 };

int odds = std::count_if(begin(nums), end(nums), isOdd);
```

```
[](){} // a valid lambda
```

Capture clause []

Parameters ()

Body {}

Wait, What Is a Lambda Really?

Compiler generates an anonymous function object

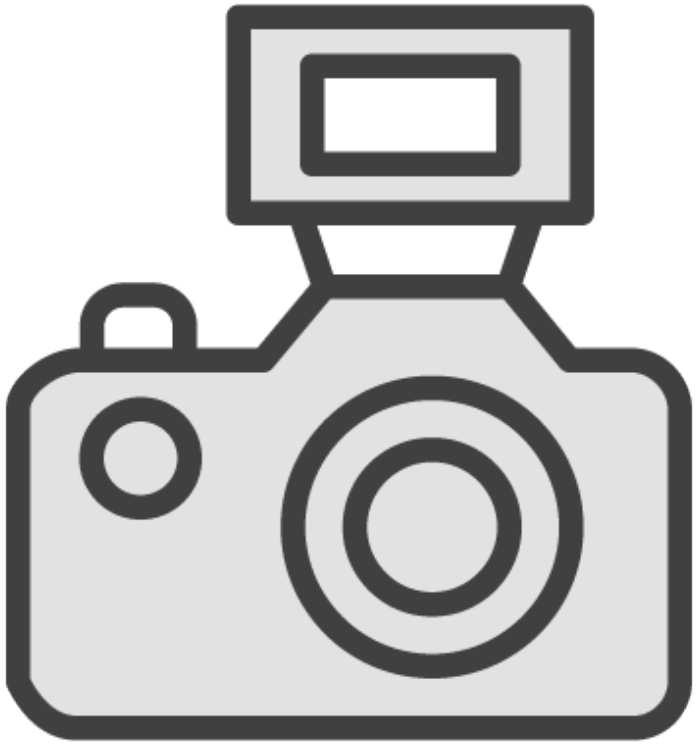
Overrides (operator

Parameters in the (), Return type after the ->

Member variables

Controlled by capture clause, const by default

The Capture



Empty [] – captures nothing, use only function parameters

[x,y] – capture x and y by value

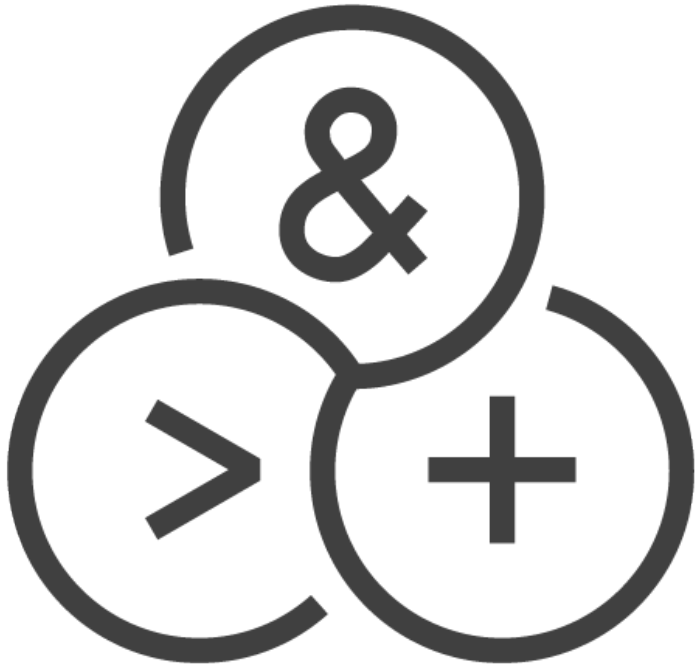
- Copies are made
- Lambda can be used when x and y have gone out of scope

[&x,&y] – capture x and y by reference

- No copies, changes affect the originals
- Dangling references may be an issue

[x=a+1,y=std::move(b)] – alias or move capture

- Useful when you need it



[=] – copy "everything" by value

- Actually it's everything used in the body of the lambda

[&] – copy "everything" by reference

- Again, only what's used

Mutable

- Allows you to change values captured by reference

How to Capture?

Lambda not stored, capture by value/reference

Lambda stored, capture by value

Use the “everything” notation

Return Value

Lambdas may return a value

Only a return statement in the lambda: return type inferred by compiler

Compiler can't infer: developer specifies return type

```
[](int n) -> double {... }
```

Parameters

What to take?

Generally imposed by the place you are using it

Like writing any other predicate

Syntactic Sugar

You could use function objects (functors) wherever you can use lambdas

And in many cases (eg standard algorithms like sort) you can use a function pointer

But

- We just didn't
- And now we do!

Lambdas keep the code where it is used

- For readability
- For expressivity (show your intent)
- For confidence no-one else uses this, so you can change it

Summary

Lambdas are “only” syntactic sugar

- But they change everything

Lambdas make for_each and other Standard Library functions suddenly usable

They open the door for interesting parallel and functional work

C++ lambdas offer more control than other languages

- Capturing by value/reference