Lambdas in C++ 11

Quick Poll!

Who here has used a C++11 compatible compiler before?

Why Lambda?

- Locally Declare Functions
- Pass functions as parameters (like functors)
- Use variables outside the scope of lambda
- Works well with STL

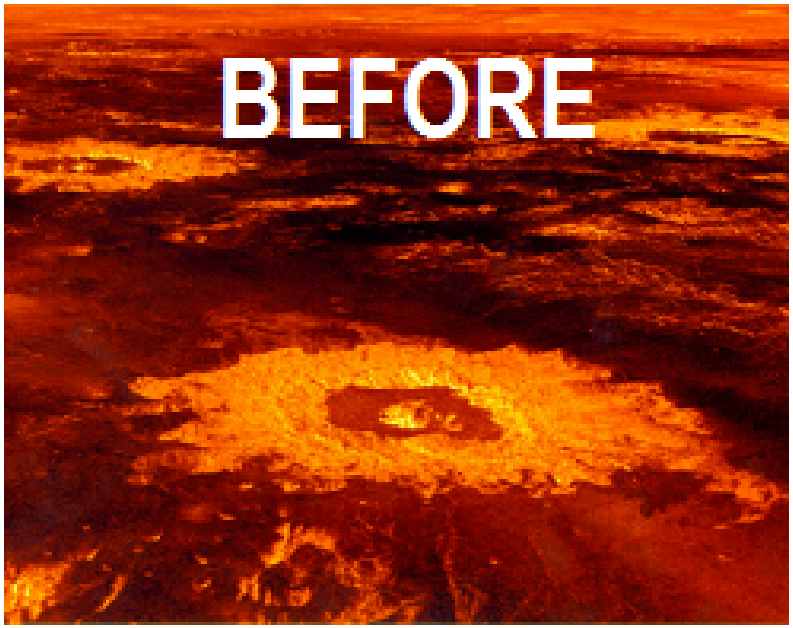
Response to Lambda

I've started using lambda functions in production code . . . shortening code. . . improving unit tests. . . replacing what could previously have only been accomplished with macros. [1]

Alex Allain

1 *http://www.cprogramming.com/c++11/c++11-lambda-closures.html*

Life Before Lambda in C++



Why?

- Functors and similar operations cumbersome to use
- Overkill to declare and only use a function once [a]

---

[a] *Stack Overflow Moderator, Flexo*
http://stackoverflow.com/questions/7627098/what-is-a-lambda-expression-in-c11

Life Before Lambda: Example

Utilizing for-each without Lambda Functions

```
#include <algorithm>
#include <vector>

namespace {
  struct f {
    void operator()(int val) {
      // do something
    }
  };
}

void func(std::vector<int>& v) {
  f f;
  std::for_each(v.begin(), v.end(), f);
}
```

Life Before Lambda: Example

One might be tempted to use this code: [2]

Locally declared:

```
void func2(std::vector<int>& v) {
  struct {
    void operator()(int val) {
       // do something
    }
  } f;
  std::for_each(v.begin(), v.end(), f);
}
```

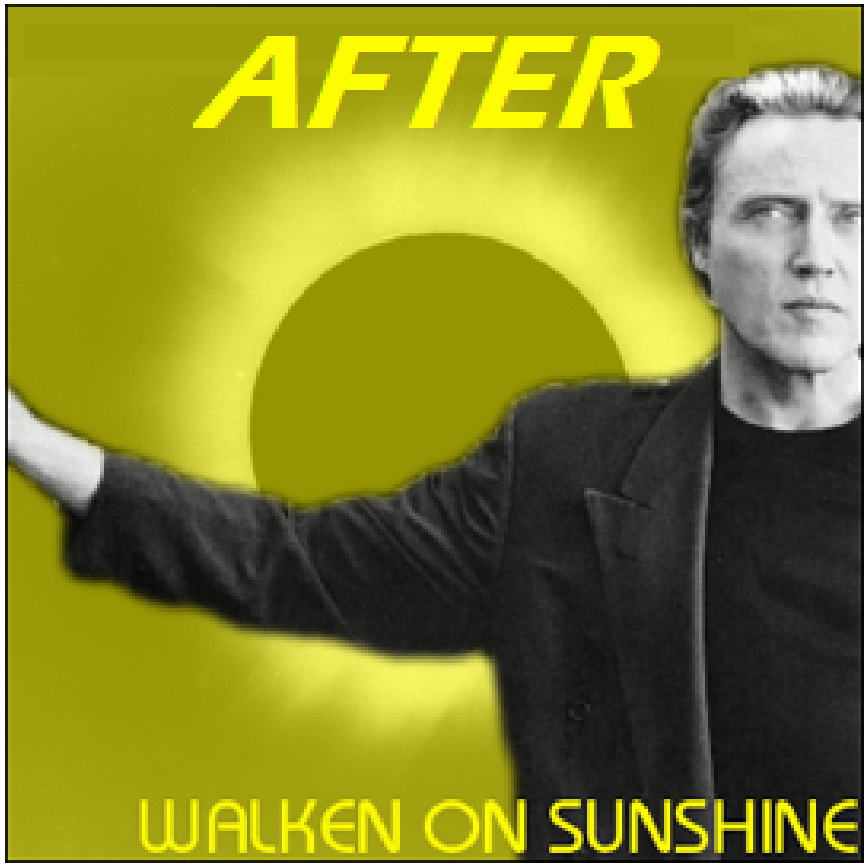
However, our struct 'f' cannot be passed to a template function!

---

2 *Stack Overflow Moderator, Flexo*
http://stackoverflow.com/questions/7627098/what-is-a-lambda-expression-in-c11

Introducing C++11 Lambdas!

Quick and easy function inlining!



Life With Lambda

Use of lambdas: For ages, people have complained about having to write functions or (better) function objects for use as operations, such as Cmp above, for standard library (and other) algorithms. This was especially painful to do if you wrote large functions (don't) because in C++98 you could not define a local function object to use as an argument; now you can. However, lambdas allows us to define operations "inline:" - Bjarne Stroustrup [3]

---

3 *http://www.stroustrup.com/C++11FAQ.html#lambda*

Lambda syntax

Constructing a Lambda function

```
lambda_type my_func =
    [capture_specification] (parameters) ->
        return_type {function_body}
```

- The function name is not required, in the case of anonymous functions.
- The capture specification field can be left empty if desired.
- Parameters and return type are not necessary in most cases:
  1. Some compilers require a return type.
  2. Other compilers will default to void.

Lambda syntax

Example:

```
#include <iostream>

using namespace std;

int main()
{
    auto func = [] () { cout << "Hello world"; };
    func(); // now call the function
}
```

Above we see a lambda function being defined and used.

Life With Lambda: Example

Here is the previous problem, resolved cleanly and effectively[4] thanks to the use of a lambda function.

For_each with Lambda

```
void func3(std::vector<int>& v) {
  std::for_each(v.begin(), v.end(),
        [](int val) { /* do something here*/ });
}
```

Stack Overflow Moderator, Flexo

http://stackoverflow.com/questions/7627098/what-is-a-lambda-expression-in-c11

The Community Responds

Lambdas are syntactic sugar for functor classes . . . it aids in the readability of code (having your functor declared inline where it is used). - Terry Mahaffey

[In my opinion], the most important thing about lambda's is it keeps related code close together. - R Samuel Klatchko

The Community Responds

There's no performance benefit per se, but the need for lambda came as a consequence of the wide adoption of the STL and its design ideas. - Fabio Ceconello

. . . is it just a neat development perk open for abuse by poor coders trying to look cool? - LoudNPossiblyWrong

Variable Capture

They can capture contexts, and they do so by name and then they can take those contexts elsewhere and execute.[5]

Variable Capture Modes:

- [=] Set default capture mode to capture by value. (Data duplicated/Copy constructor invoked)
- [&] Set default capture mode to capture by reference. (No data duplication, function side-effects can persist)

---

5 *Cubbi* *http://stackoverflow.com/questions/3018626/what-is-the-motivation-behind-c11-lambda-expressions*

Variable Capture: Examples

Examples

- [variable_name] Capture individual variables.
  (In this case, by value)
- [&variable_name] Same as above, but by reference.
- [=, &var_ref] Capture all inferred variables by value, except for var_ref which is captured by reference.
- [ ] The function body captures no variables.

Variable Captures and Lambdas

Capturing Other Lambda Functions

- When capturing with [=] or [&] you can also capture other lambda functions declared in the same scope.
- Any standard functions declared directly inside a namespace is also available inside of lambda bodies.
- This allows the easy construction of higher order functions.

Variable Capture Example

Capturing By-Value:

```
#include <iostream>
using namespace std;

int main() {
    string input;
    cout << "enter a number: ";
    cin >> input;

    auto lambda_isNeg = [input] {
        return input < 0 ? "negative" : "positive"
    }
    cout << "you entered a " << lambda_isNeg;
    cout << "number" << endl;
    return 0;
}
```

Overall

Lambdas are

- Well accepted by the community
- useful for adding to code expressiveness
- and very powerful due to their flexible variable capture

Concluding Lambdas

Thank you for listening, we hope we have enlightened you to one of the new, useful C++11 features.