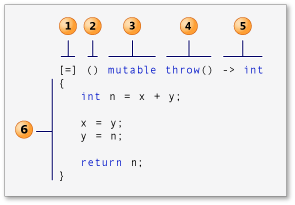
In C++11 and later, a lambda expression—often called a *lambda*—is a convenient way of defining an anonymous function object (a *closure*) right at the location where it is invoked or passed as an argument to a function. Typically lambdas are used to encapsulate a few lines of code that are passed to algorithms or asynchronous methods.

This illustration shows the parts of a lambda:



1. capture clause (Also known as the lambda-introducer in the C++ specification.)
2. parameter list Optional. (Also known as the lambda declarator)
3. mutable specification Optional.
4. exception-specification Optional.
5. trailing-return-type Optional.
6. lambda body.

Reference:

<https://stackoverflow.com/questions/220658/what-is-the-difference-between-a-closure-and-a-lambda>

<https://www.cprogramming.com/c++11/c++11-lambda-closures.html>

<https://docs.microsoft.com/en-us/cpp/cpp/lambda-expressions-in-cpp?view=vs-2019>

<https://stackoverflow.com/questions/7627098/what-is-a-lambda-expression-in-c11>

<https://www.geeksforgeeks.org/lambda-expression-in-c/>

<https://en.cppreference.com/w/cpp/language/lambda>

<http://scottmeyers.blogspot.com/2013/05/lambdas-vs-closures.html>