# Lambda functions

Syntax:

**[** capture-list **]** **(** params **)** **{** body **}**

Lambdas are useful for example in functions that as an argument, take a comparison function (known as “Comparator”).

For example, std::sort function has this definition:

template< class RandomIt, class Compare >  
void sort( RandomIt first, RandomIt last, Compare comp );

If we have objects of a class Human, and would like to sort them, how would std::sort know how to sort those objects?

We have to give it a function, which will explain how to sort (by age or height etc).

In C++03, it was done by the Functor.

A functor is class\struct which defines the operator()

C++03 Example:

Our class that we want to sort:

class Human

{

public:

int age;

int height;

std::string name;

};

struct Human\_Sort\_Function //our Functor

{

bool operator()(const Human &one, const Human &two)

{

return one.age < two.age;

}

};

Usage:

std::vector<Human> humans; //Lets assume we filled vector with 100 humans

Human\_Sort\_Function functor;

std::sort(humans.begin(), humans.end(), functor);

In C++11 using lambdas this example would look like this:

std::vector<Human> humans; //Lets assume we filled vector with 100 humans

std::sort(humans.begin(), humans.end(),[]( const Human &one, const Human &two) { return one.age < two.age; } );

No functor needed! All in one line! :)

**Lambdas layout in memory:**

Given that we have a function:

void func()

{

int a = 5;

std::string word = "Hi there";

auto our\_lambda = [&word, a](int nr) { return nr + a; };

int result = our\_lambda(10); //result = 15

}

This lambda will be created by compiler like this:

struct Compiler\_Generated\_Lambda

{

std::string &word;

int a;

int operator() (int nr)

{

return nr + a;

}

Compiler\_Generated\_Lambda(std::string &word, int a) : this->word(word), this->a(a) {} //constructor

};

This is how lambdas are created. So for compiler it is the same as functor, but for users, it is a syntactic sugar, that helps with readability and requires less code.