


Callables

- Many algorithms can be parametrized with callables.
- Callable
 - Something that behave like a function
 - ➔ Function, function object, or lambda function

Specialties

-  ▪ In order to modify the elements of a container, you have to use references.
- Predicates are special callables that returns boolean values.

Callable

Functions

- Simple callable units
- Cannot have state

```
void square(int& i){  
    i = i * i;  
}
```

```
std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
std::for_each(myVec.begin(), myVec.end(), square);
```

callableUnitFunction.cpp

Callable

Function objects

- are objects that behave like functions.
- have an overloaded call operator.
- can have state.

```
struct Square{  
    void operator()(int& i){  
        i = i * i;  
    }  
};  
std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
std::for_each(myVec.begin(), myVec.end(), Square());
```



In order to use function objects, you have to instantiate them.

`callableUnitFunctionObject.cpp`

Callable

C++ has predefined function objects

- They are defined in the header `<functional>`.
- Help to adjust the default behavior of STL containers.

```
std::map<int, std::string> myDefaultMap; // std::less<int>  
std::map<int, std::string, std::greater<int>> mySpecialMap;
```

Kind of Function Object	Operations
Arithmetic operations	plus, minus, multiplies, divides, modulus, negate
Comparison operations	equal_to, not_equal_to, less, greater, less_equal, greater_equal
Logical operations	logical_not, logical_and, logical_or
Bitwise operations	bit_and, bit_or, bit_xor

Callables

Lambda expressions

- define their functionality in place.
- offer a high optimization potential.
- should be concise.

```
std::vector<int> myVec{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

```
std::for_each(myVec.begin(), myVec.end(), [](int& i){i = i * i;});  
std::for_each(myVec.begin(), myVec.end(), [](int i){  
    std::cout << i << " ";  
});
```



Lambda functions should be the first choice for callables.

`callableUnitLambda.cpp`