Relaxing the requirements on standard library algorithms predicates and functors

Document #:	P0XXXR0
Date:	2018-03-13
Project:	Programming Language C++
Audience:	LEWG

Reply-to: Georg Rudoy <georgii.rudoi@phystech.edu>

1 Revision history

- R0 — Initial draft

2 Abstract

This paper proposes a (strictly relaxing) change in the definition of several algorithms in the standard library. Various algorithms like `std::any_of`, `std::find_if` and the likes are currently defined in terms of function call syntax on the predicate/function, forbidding passing pointers to members where they might have sufficed.

3 Motivation

```
struct Standard
{
    bool isCurrent;
};

vector<Standard> stds;

auto hasCurrent = any_of(stds.begin(), stds.end(),
        &Standard::isCurrent);                              // nope
auto hasCurrent = any_of(stds.begin(), stds.end(),
        [](auto&& std) { return std.isCurrent; });          // ok


class Employee
{
public:
```

```
    void fire();
    bool shouldBeFired() const;
};

vector<Employee> emps;

auto poorFellow = find_if(emps.begin(), emps.end(),
        &Employee::shouldBeFired);                              // nope
auto poorFellow = find_if(emps.begin(), emps.end(),
        [](auto&& emp) { return emp.shouldBeFired(); });        // ok

for_each(emps.begin(), emps.end(), &Employee::fire);            // nope
for_each(emps.begin(), emps.end(),
        [](auto&& emp) { emp.fire(); });                        // ok
```

4 The fix

Replace `pred(*i)` and the likes with `std::invoke(pred, *i)` and the likes.

5 Proposed wording

In **[alg.all_of]/2**:

> *Returns*: `true` if `[first, last)` is empty or if ~~`pred(*i)`~~ std::invoke(pred, *i) is `true` for every iterator `i` in the range `[first, last)`, and `false` otherwise.

In **[alg.any_of]/2**:

Ditto.