# Improving the Return Value of Erase-Like Algorithms

Document #: D0xxxR0
Date: 2017-05-18
Project: Programming Language C++
Library Evolution Working Group
Reply-to: Marc Mutz <marc.mutz@kdab.com>

## 1 Introduction

We propose to change the return type of [N4600] `erase()` and `erase_if()` algorithms, as well as `forward_list::remove()` and `remove_if()` from `void` to `size_t`, returning the number of elements removed.

## 2 Motivation and Scope

Alexander Stepanov, in his A9 courses[A9], teaches us not to throw away useful information, but instead return it from the algorithm.

With that in mind, look at the following example:

```
std::forward_list<std::shared_ptr<T>> fl = ...;
erase(fl, nullptr);
```

Did `erase()` erase anything? We don't know. The only way we *can* learn whether the algorithm removed something is to check the size of the list before and after the algorithm run. For most containers, that is a valid option, and fast. All `size()` methods of STL containers are $O(1)$ these days.

But `std::forward_list` has no `size()`...

We therefore propose to make the algorithms return the number of removed elements. While it is only really necessary for `forward_list`, we believe that consistency here is more important than minimalism.

Returning the number of elements also enables convenient one-line checks:

```
if (erase(lf, nullptr)) {
    // erased some
}
```

## 3 Impact on the Standard

Minimal. We propose to change the return value of library functions from `void` to `size_t`. Existing users expecting no return value can continue to ignore it.

## 4 Proposed Wording

### 4.1 Changes to [N4659]

In section [**forwardlist.overview**]:

- in paragraph 3, change the `remove()`, `remove_if()` and `unique()` return types from `void` to `size_t` (four instances).

In section [**forwardlist.ops**]:

- after paragraphs 11 and 15, change the `remove()`, `remove_if()` and `unique()` return types from `void` to `size_t` (four instances).
- after paragraphs 12 and 16, add new paragraph each:

  *Returns:* The number of elements erased.

In section [**list.overview**]:

- in paragraph 2, change the `remove()`, `remove_if()` and `unique()` return types from `void` to `size_t` (four instances).

In section [**list.ops**]:

- after paragraphs 14 and 18, change the `remove()`, `remove_if()` and `unique()` return types from `void` to `size_t` (four instances).
- after paragraphs 15 and 19, add new paragraph each:

  *Returns:* The number of elements erased.

### 4.2 Changes to [N4600]

In section [**container.erasure.erase_if**]:

- replace all `void` return types with `size_t`
- change paragraph 2 to

  *Effects:* Equivalent to:

```
auto it = remove(c.begin(), c.end(), value);
auto res = size_t(distance(it, c.end()));
c.erase(it, c.end());
return res;
```

- add new paragraph after each of paragraphs 2, 4, and 6:

  *Returns:* The number of elements erased.

- in paragraph 4, insert `return` between "Equivalent to:" and "`c.remove_if(...`".
- change paragraph 4 to

  *Effects:* Equivalent to:

```
+ size_t res = 0;
  for (auto i = c.begin(), last = c.end(); i != last; ) {
    if (pred(*i)) {
      i = c.erase(i);
+     ++res;
    } else {
      ++i;
    }
  }
+ return res;
```

In section [**container.erasure.erase**]:

- replace all `void` return types with `size_t`
- change paragraph 2 to

  *Effects:* Equivalent to:

```
auto it = remove(c.begin(), c.end(), value);
auto res = size_t(distance(it, c.end()));
c.erase(it, c.end());
return res;
```

- add new paragraph after each of paragraphs 2 and 4:

  *Returns:* The number of elements erased.

- in paragraph 4, insert `return` between "Equivalent to:" and "`erase_if(...`".

## 5 References

[A9] Alexander Stepanov *et al.*: *Four Algorithmic Journeys / Efficient Programming With Components / Programming Conversations*
https://www.youtube.com/user/A9Videos/playlists?view=1

[N4600] Geoffrey Romer (editor): *Working Draft, C++ Extensions for Library Fundamentals, Version 2*
http://open-std.org/JTC1/SC22/WG21/docs/papers/2016/n4600.html

[N4659] Richard Smith (editor). *Working Draft, Standard for Programming Language C++*
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4659.pdf