Loop Exit Blocks

A proposal for C++20

Alan Talbot
cpp@alantalbot.com

C++11/14/17

```
auto it = get_begin(. . .);
auto end = get_end(. . .);
for (; it != end; ++it)
{
    if (some_condition(*it)) break;
    do_something(*it);
}
if (it == end)
    do_stuff();
else
    do_something_else(*it);
```

C++11/14/17

```
auto it = get_begin(. . .);
auto end = get_end(. . .);
for (; it != end; ++it)
{
    if (some_condition(*it)) break;
    do_something(*it);
}
if (it == end)
    do_stuff();
else
    do_something_else(*it);
```

C++20?

```
for (auto it(get_begin(. . .)),
     end(get_end(. . .));
     it != end; ++it)
{
    if (some_condition(*it)) break;
    do_something(*it);
}
if final
    do_stuff();
if break
    do_something_else(*it);
```

C++11/14/17

```
bool early = false;
while (some_condition())
{
    . . .
    if (test1()) { early = true; break; }
    . . .
    if (test2()) { early = true; break; }
    . . .
    if (test3()) { early = true; break; }
    . . .
}
if (early)
{ . . . }
else
{ . . . }
```

C++11/14/17

```
bool early = false;
while (some_condition())
{
    . . .
    if (test1()) { early = true; break; }
    . . .
    if (test2()) { early = true; break; }
    . . .
    if (test3()) { early = true; break; }
    . . .
}
if (early)
{ . . . }
else
{ . . . }
```

C++20?

```
while (some_condition())
{
    . . .
    if (test1()) break;
    . . .
    if (test2()) break;
    . . .
    if (test3()) break;
    . . .
}
if break
{ . . . }
if final
{ . . . }
```

C++11/14/17

```
element_type* found = nullptr;
for (auto&& element : container)
{
    if (some_condition(element))
    {
        found = &element;
        break;
    }
    do_something(element);
}
if (found)
    do_something_else(*found);
else
    do_stuff();
```

C++11/14/17

```
element_type* found = nullptr;
for (auto&& element : container)
{
    if (some_condition(element))
    {
        found = &element;
        break;
    }
    do_something(element);
}
if (found)
    do_something_else(*found);
else
    do_stuff();
```

C++20?

```
for (auto&& element : container)
{
    if (some_condition(element)) break;
    do_something(element);
}
if break
    do_something_else(element);
if final
    do_stuff();
```

C++11/14/17

```cpp
auto find_and_do = [&container, ...]()
    ->element_type*
{
    for (auto&& element : container)
    {
        if (some_condition(element))
            return &element;
        do_something(element);
    }
    return nullptr;
};

if (auto found = find_and_do())
    do_something_else(*found);
else
    do_stuff();
```

C++20?

```cpp
for (auto&& element : container)
{
    if (some_condition(element)) break;
    do_something(element);
}
if break
    do_something_else(element);
if final
    do_stuff();
```

C++11/14/17

```
for (auto i = container.begin();
          i != container.end(); ++i)
    if (some_condition(*i))
    {
        container.erase(i);
        break;
    }
assert(what???);
```

C++11/14/17

```
for (auto i = container.begin();
          i != container.end(); ++i)
    if (some_condition(*i))
    {
        container.erase(i);
        break;
    }
assert(what???);
```

C++20?

```
for (auto i = container.begin();
          i != container.end(); ++i)
    if (some_condition(*i))
    {
        container.erase(i);
        break;
    }
if final
    assert(false);
```

C++11/14/17

```
for (auto i = container.begin();
          i != container.end(); ++i)
    if (some_condition(*i))
    {
        container.erase(i);
        break;
    }
assert(what???);
```

C++20?

```
for (auto i = container.begin();
          i != container.end(); ++i)
    if (some_condition(*i))
        break;
if break
    container.erase(i);
if final
    assert(false);
```

C++11/14/17

```
for (auto& x : table)
    for (auto& y : x)
        for (auto& z : y)
            if (some_condition(z))
            {
                do_something(z);
                goto DONE;
            }
DONE:
```

C++11/14/17

```
for (auto& x : table)
    for (auto& y : x)
        for (auto& z : y)
            if (some_condition(z))
            {
                do_something(z);
                goto DONE;
            }
DONE:
```

C++20?

```
for (auto& x : table)
    for (auto& y : x)
        for (auto& z : y)
            if (some_condition(z))
            {
                do_something(z);
                break;
            }
        if break break;
    if break break;
```

`on_complete`	`on_break`
`on complete`	`on break`
`oncomplete`	`onbreak`
`catch complete`	`catch break`
`if complete`	`if break`
`if final`	`if break`

Alan Talbot
cpp@alantalbot.com