Wording for [[fallthrough]] attribute.

Document No.: P0188R0
Revises: P0068R0 In Part
Project: Programming Language C++ - Evolution
Author: Andrew Tomazos <andrewtomazos@gmail.com>
Date: 2016-01-03

Summary

Wording for the [[fallthrough]] attribute described in P0068R0 is proposed for application to the C++17 working draft. [[fallthrough]] captures an intent to use the feature of “falling-through” from one case block to the next, in a switch statement. It has heavy use in existing practice. Kona EWG voted SF=15, F=5, N=0, A=0, SA=0 in favor of [[fallthrough]] from P0068R0. See P0068R0 for detailed motivation/rationale.

Wording

7.6.8 Fallthrough attribute [dcl.attr.fallthrough]

1. A null statement marked with the attribute-token **`fallthrough`**, is a *fallthrough statement*. The fallthrough attribute-token shall appear at most once in each attribute-list, with no attribute-argument-clause.
2. A fallthrough statement may appear within an enclosing switch statement, on some path of execution immediately between a preceding statement and a succeeding case-labeled statement.
3. [Note: If an implementation would have otherwise issued a warning about implicit fall through on a path of execution immediately after a fallthrough statement, it is encouraged not to. -- end note]

Example

Compiled with an implementation-defined implicit fallthrough warning enabled:

```
switch (n) {
case 22:
case 33:  // OK: no statements between case labels
  f();
case 44:  // WARNING: no fallthrough statement
  g();
  [[fallthrough]];
```

```
case 55:  // OK
  if (x) {
    h();
    break;
  }
  else {
    i();
    [[fallthrough]];
  }
case 66:  // OK
  p();
  [[fallthrough]]; // WARNING: fallthrough statement out-of-place
  q();
case 77:  // WARNING: no fallthrough statement
  r();
}
```

FAQ

1. Why does [[fallthrough]] need a trailing semi-colon? Why doesn’t it annotate the case label?

On 2015–09–09, at 7:41 AM, Richard Smith <richard@metafoo.co.uk> wrote:

> The argument when we designed [[fallthrough]] was that [[fallthrough]] doesn't notionally appertain to the label -- it appertains to the *preceding* sequence of labelled statements. Note that when you have a sequence of case labels with no intervening statements, it allows fallthrough through all of them, so it doesn't meaningfully apply to just one label. Also observe Example 3, where the fallthrough within the 'case 55:' block is not even immediately lexically preceding a case label.
>
> We viewed [[fallthrough]] as being more of a flow control keyword (being provided as an extension) than a source annotation, and from that perspective it made sense for it to be a new kind of statement (like a break statement or continue statement). (This also allows source compatibility with existing systems that already have such a keyword -- see for instance the "__fallthrough;" statement provided by MS SAL, which can be implemented with this proposal as "#define __fallthrough [[fallthrough]]", but cannot be implemented with a label attribute.)

On 2015–09–09, at 8:48 AM, Andrew Tomazos <andrewtomazos@gmail.com> wrote:

> Consider the following:
>
> ```
> switch (n) {
> case 2:
>   if (c1) {
> ```

```
      f();
      break;
    } else if (c2) {
      g();  // WARNING: no fallthrough statement
    } else if (c3) {
      h();
      break;
    } else if (c4) {
      g();
      h();
      [[fallthrough]];
    }
  case 3:
    h();
}
```

This can be addressed with this:

```
  switch (n) {
  case 2:
    if (c1) {
      f();
      break;
    } else if (c2) {
      g();
      break; // <---- bug fixed
    } else if (c3) {
      h();
      break;
    } else if (c4) {
      g();
      h();
      [[fallthrough]];
    }
  case 3:
    h();
  }
```

or this:

```
  switch (n) {
  case 2:
    if (c1) {
      f();
      break;
    } else if (c2) {
      g();
      [[fallthrough]]; // <--- nope, intentional
```

```
    } else if (c3) {
      h();
      break;
    } else if (c4) {
      g();
      h();
      [[fallthrough]];
    }
  case 3:
    h();
  }
```

Had we specified fallthrough as appertaining to the label, and not as statements, this bug would have been missed. Or rather, the programmer would not be able to express the fallthrough intent of each block individually within the if-else chain as shown above.

fallthrough appertains to points on one or more paths of execution. It can be thought of as an assertion statement that the next thing that will happen at run-time is the next case block will be executed. (This assertion is checked statically at compile-time.)