n2859 – break break

Document #n2859
Date 10/15/2021
Author Eskil steenberg (eskil@quelsolaar.com)

Description:

Break and continue statements are very useful for flow control, but often they are slightly too limited, becaus they can only exit one loop or switch. Therefore I propose the ability to have multiple break statements in a row, or one or more break statements followed by a continue.

Consider:

```
for(i = 0; i < n; i++)
      for(j = 0; j < n; j++)
            if(something(i, j))
                  goto end;
end :
```

Many users want to avoid using goto, and would rather prefer to use a break. However a break can only exit one loop/switch. Trying to avoid a goto, can produce some awkward code:

```
for(i = 0; i < n; i++)
{
      for(j = 0; j < n; j++)
            if(something(i, j))
                  break;
      if(j < n)
            break;
}
```

Compilers are able to detect that these are identical:
`https://godbolt.org/z/K1KshY91r`
`https://godbolt.org/z/fq3W9Ezos`

Instead I propose the ability to make a multi-break statement:

```
for(i = 0; i < n; i++)
     for(j = 0; j < n; j++)
          if(something(i, j))
               break break;
```

I also propose that break statements can end with a continue statement. An example of this would be:

```
for(i = 0; i < n; i++)
{
     for(j = 0; j < n; j++)
          if(something(i, j))
               break continue;
     something_else();
}
```

That would be equivalent to:

```
for(i = 0; i < n; i++)
{
     for(j = 0; j < n; j++)
          if(something(i, j))
               break;
     if(j < n) /* did we break? */
           continue;
     something_else();
}
```

Discussion:

-Why not just use goto?

goto has (somewhat undeservedly) a reputation of being bad style. Many users avoid goto, and some style guides even forbid its use. Even if one does accept the use of goto, it has the disadvantage that a goto label can be placed anywhere. break and continue statements are localized and therefore easier to read, and less likely to be abused.

-Why not use a constant value to denote the number of loops/switches to exit? Along the lines of break[2]; or continue(2);

A value would suggest that the break statement could have dynamic target. Multiple breaks is clearly not dynamic. Having one or more break followed by a continue is also clearer than having a continue with a index. Using multiple breaks doesn't break existing code, and doesn't add new keywords that pollute the namespace. It is a feature that solves a common problem, with a small change.

Proposed changes:

6.8.6 Jump statements

Syntax

1 jump-statement:

`goto` identifier `;`

one or more opt break `continue ;`

additional opt break `break ;`

`return` expressionopt `;`

6.8.6.2 The `continue` statement

Constraints

1 A `continue` statement shall appear only in or as a loop body. For each preceding break, the statement must appear in an additional loop body.

Semantics

2 A `continue` statement causes a jump to the loop-continuation portion of the smallest an enclosing iteration statement; that is, to the end of the loop body. More precisely, in each of the statements unless the `continue` statement shown is in an enclosed iteration statement (in which case it is

interpreted within that statement), it is equivalent to `goto contin;`
A `continue` statement without preceding break jumps to the smallest enclosing iteration statement. For each preceding break, the jump reaches one further iteration or `switch` statement out.

```
while (/* ... */) {
    while (/* ... */) {
    /* ... */
    break continue;
    }
/* ... */
contin:
}

while (/* ... */) {
    switch (/* ... */) {
    /* ... */
    break continue;
    }
/* ... */
contin:
}
```

6.8.6.3 The `break` statement
Constraints
1 A `break` statement shall appear only in or as a switch body or loop body. For each preceding break, the statement must appear in an additional switch body or loop body.

Semantics
2 A single `break` statement terminates execution of the smallest enclosing `switch` or iteration statement. For each additional break contained in the statement, an additional enclosing `switch` or iteration statement is terminated starting from the innermost going outwards.

Question for the WG14

Does the wg14 want something along the lines of n2859, into c23?

References:

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n3879.pdf