An *evaluation* is the execution of a statement, expression, constructor, destructor, or initialization of an object (including unnamed temporary objects). [*This definition is for our benefit only; is not needed in the TS because the word “evaluation” appears in the standard as something that can be sequenced.*]

Alternatives names for *horizontally matched:*

* *aligned*
* *vector aligned, vector matched, (have) vector correspondence*
* *horizontally aligned, horizontally matched, (have) horizontal correspondence*
* *laterally aligned, laterally matched, (have) lateral correspondence*

Two evaluations, Ei and Ej of the same statement, expression, object initialization (including initialization of unnamed temporary objects), or destructor invocation E, in two different vector applications i and j of F might be *horizontally matched*, according to the following rules:

Given evaluations Xi and Xj, such that Xi is horizontally matched with Xj,

* If X is a selection statement, conditional expression, or an evaluation of (built-in) operator && or operator || with condition C, then Ci is horizontally matched with Cj. If both evaluations select the same sub-statement or sub-expression Y, then Yi is horizontally matched with Yj.
* Otherwise, if X is an iteration statement, then the evalutions of the *for-init-*statements Vi and Vj (for for loops) are horizontally matched and the first evaluations of the condition, Ci and Cj (for for and while loops) are horizontally matched. Each evaluation of the remaining subexpressions (including subsequent evaluations of the condition) and substatements in application i is horizontally matched with the corresponding subexpressions and substatements in application j, where *corresponding* refers to the ordered sequence of iterations, irrespective of the values of the loop control variables. If the applications execute different numbers of iterations, the common iterations are horizontally matched and the remainder are not.
* [Some rule(s) about gotos, longjmp, throw, and unstructured switch goes here]
* Otherwise, if Y is a subexpression, substatement, object initialization, or destructor invocation within X, then Yi is horizontally matched with Yj.

For distinct evaluations of Xj, Xj, Yj, and Yj applications i and j, where i would precede j in the sequential order of applications, *wavefront application* of F provides the following sequencing guarantees:

* If Xi is horizontally matched with Xj and Xj is sequenced before Yj, then Xi is sequenced before Yj.
* If Yi is horizontally matched with Yj and Xi is sequenced before Yi, then Xi is sequenced before Yj.

for\_loop(par, 0, 4, [&](int x){

if (x % 2)

f(x);

else

g(x);

for (int y = x / 2; y < 2; ++y)

h(y);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Code** | **F(0)** | **F(1)** | **F(2)** | **F(3)** |
| **if (x % 2)** | 0 % 2 | 1 % 2 | 2 % 2 | 3 % 2 |
| **f(x);** |  | f(1) |  | f(3) |
| **g(x);** | g(0) |  | g(2) |  |
| **for(int y = x/2;** | y = 0/2 | y = 1/2 | y = 2/2 | y = 3/2 |
| **y < 2;** | 0 < 2 | 0 < 2 | 1 < 2 | 1 < 2 |
| **h(y);** | h(0) | h(0) | h(1) | h(2) |
| **++y)** | y = 1 | y = 1 | y = 2 | y = 2 |
| **y < 2;** | 1 < 2 | 1 < 2 | 2 < 2 | 2 < 2 |
| **h(y);** | h(1) | h(1) |  |  |
| **++y)** | y = 2 | y = 2 |  |  |
| **y < 2;** | 2 < 2 | 2 < 2 |  |  |
|  |  |  |  |  |

});

Xi

Xj

Yj

Xi

Yi

Yj

Apply x = 0

if (x % 2)

f(x);

else

g(x);

h(x);

Apply x = 1

if (x % 2)

f(x);

else

g(x);

h(x);

Apply x = 2

if (x % 2)

f(x);

else

g(x);

h(x);

Apply x = 3

if (x % 2)

f(x);

else

g(x);

h(x);