Examples of operations that may use already present c APIs -

NOTE – 1.This is assumption and would require more research

2.These are not direct wraps but can be usefull

some -- @stdlib/ndarray/base/every

- every is already implemented in C and can be extended for multi-axis reductions.
- **some** is similar and can be implemented using a similar reduction approach.

includes / indexOf / lastIndexOf -- @stdlib/ndarray/base/ind2sub + @stdlib/ndarray/base/sub2ind

- Searching in ndarrays requires converting indices between different views and buffer layouts.
- ind2sub and sub2ind help translate between 1D buffer indices and multi-dimensional indices.

copyWithin -- @stdlib/ndarray/base/assign

- Copying data efficiently is handled by assign, ensuring strided data is properly transferred.

concat -- @stdlib/ndarray/base/assign + @stdlib/ndarray/base/numel

- assign can be used to copy concatenated data efficiently.
- numel helps calculate memory allocation needs before concatenation.

reduce / reduceRight -- @stdlib/ndarray/base/unary-accumulate

- Reductions accumulate results across axes, and unary-accumulate may help optimize this.

flat / flatMap -- @stdlib/ndarray/base/shape2strides

- Flattening an array is mostly about reinterpreting its memory layout without copying data.

- shape2strides helps compute how to lay out a flattened version efficiently.

sort / toSorted -- @stdlib/ndarray/base/iteration-order

- Sorting needs efficient iteration over the ndarray, which is handled by iteration-order.
- Sorting could also leverage strides2order for memory-efficient in-place sorting.

push / pop / shift / unshift -- @stdlib/ndarray/base/assign

- These operations require moving data, which can be optimized using assign.