Array Programming in Whiley

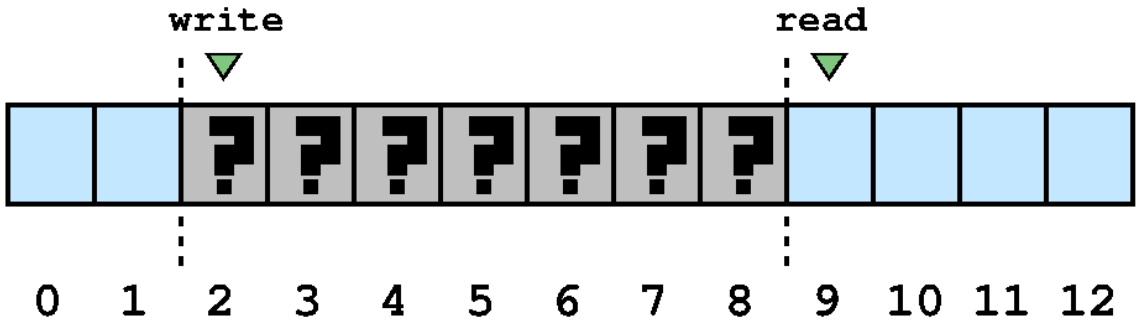
David J. Pearce

School of Engineering and Computer Science
Victoria University of Wellington

@WhileyDave
http://whiley.org
http://github.com/Whiley

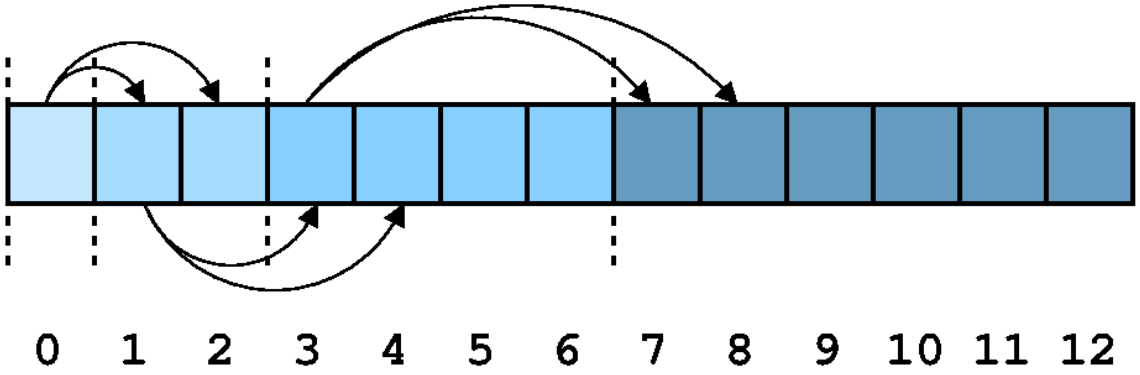
Array Programming has a **Bad Rap**

Array Programming (is being **held back**)



- Array Programming has a **bad rap** ...
- ... partly as modern languages make arrays feel **low level**
- ... and such languages also provide **attractive alternatives**

Array Programming (is **powerful**)



- Can encode **wide-range** of data structures with arrays
- No need for **references** ... and easy to manage **aliasing**!
- Easy to turn into **bits** (e.g. for the network or disk)
- Easy to **reason about** as can express key properties

Array Programming (Good for **Verification Examples**)

```
function linearSearch(int[] xs, int x) -> (int r)
// Array must contain item to find
requires some { k in 0..|xs| | xs[k] == x }
// Resulting index identifies item in array
ensures xs[r] == x:
    ...
```

- Many **verification examples** employ arrays
- Dutch National Flag problem is **widely used**
- VSCOMP'10 ($\frac{3}{5}$); COST'11 ($\frac{2}{3}$); VerifyThis'16 ($\frac{1}{3}$)

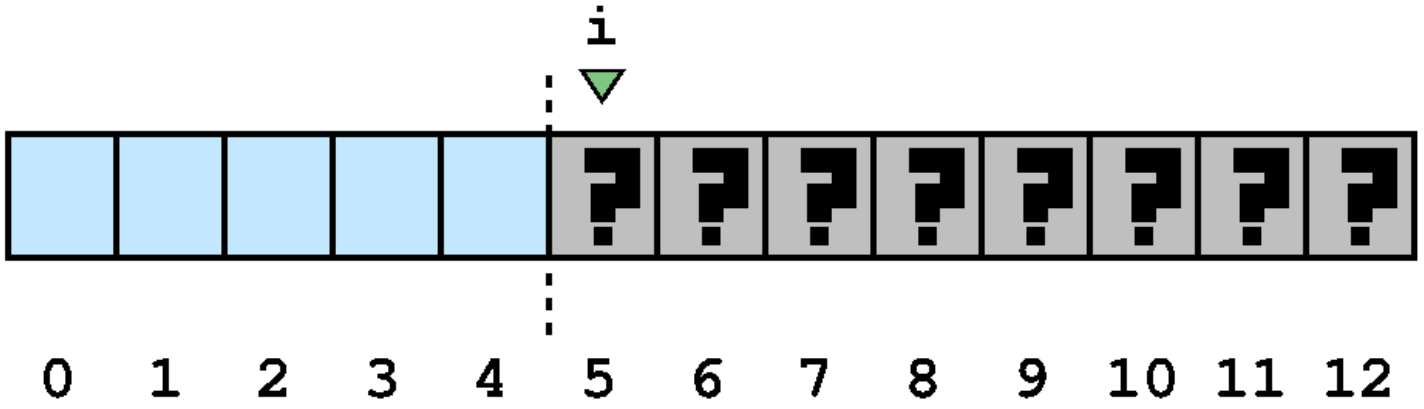
Overview: What is Whiley?

```
function max(int x, int y) -> (int z)
// result must be one of the arguments
ensures (x == z) || (y == z)
// result must be greater-or-equal than arguments
ensures (x <= z) && (y <= z):
    ...
```

- A language designed specifically to simplify **verifying software**
- Several trade offs e.g. **performance for verifiability**
  - *Unbounded Arithmetic, value semantics, etc*
- **Goal**: to statically verify functions meet their specifications

Case Study: Maximal Element

“Given an array, return the largest value contained therein.”

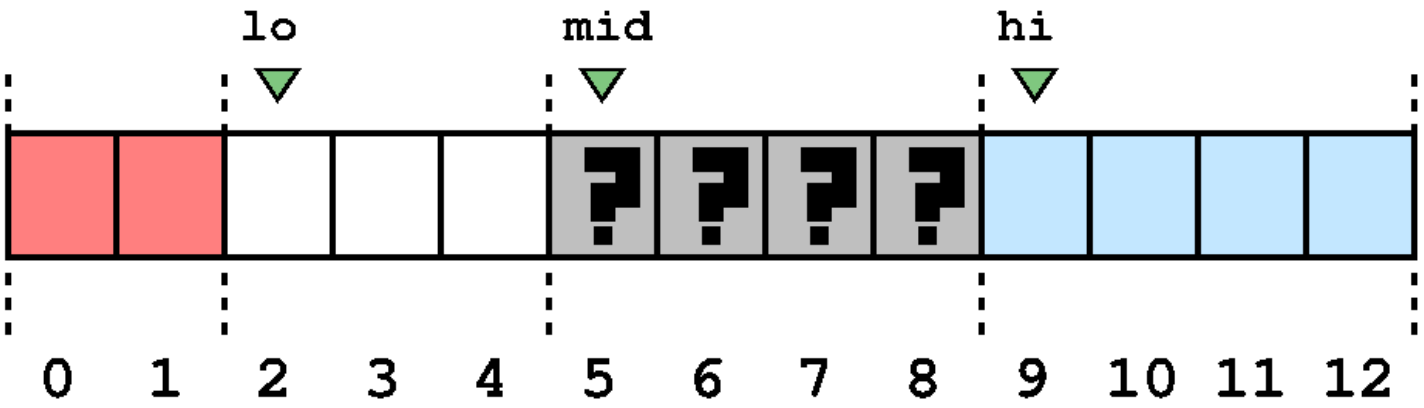


EXAMPLE: Dutch National Flag Problem

- **Problem Statement**:

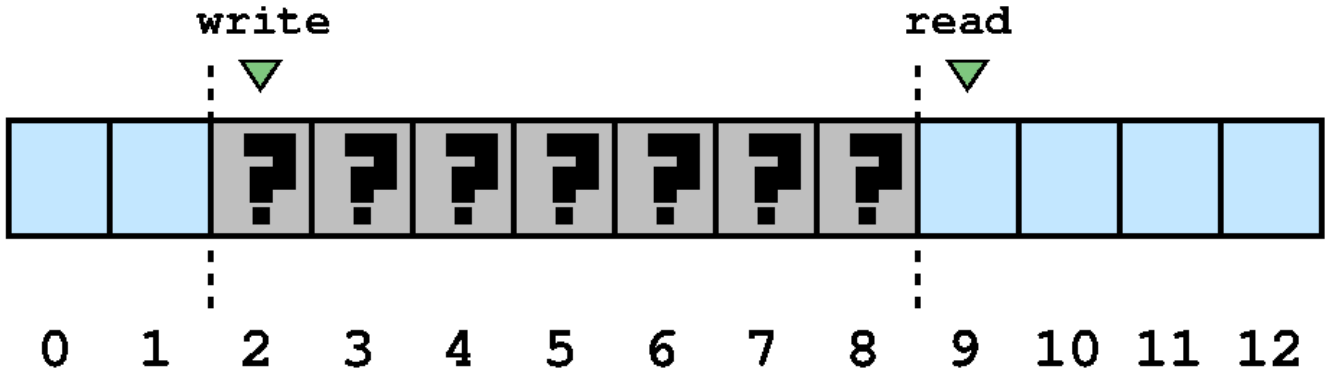
  “Given a quantity of items in three colours of the Dutch National flag, partition the items into three groups such that red items come first, then white items and, finally, blue items.”

General State



- Marker `lo` — identifies **next position** for `RED` item
- Marker `mid` — identifies **next position** for `WHITE` item
- Marker `hi` — identifies **next position** for `BLUE` item

Case Study: Cyclic Buffer

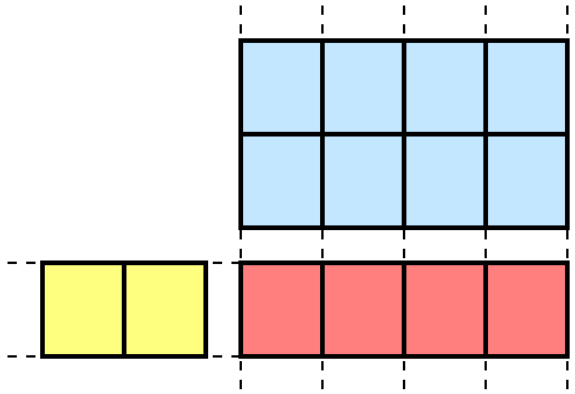



```
type Buffer is { int[] data, nat read, nat write }
// Read / write pointers within bounds
where read < |data| && write < |data|

// Buffer is empty when read and write pointers same
type EmptyBuffer is (Buffer b) where b.read == b.write

// NonFull buffer has at least one writeable space.
type NonFull is (Buffer b)
// Write cannot be immediately behind read
where ((b.write+1) % |b.data|) != b.read
```

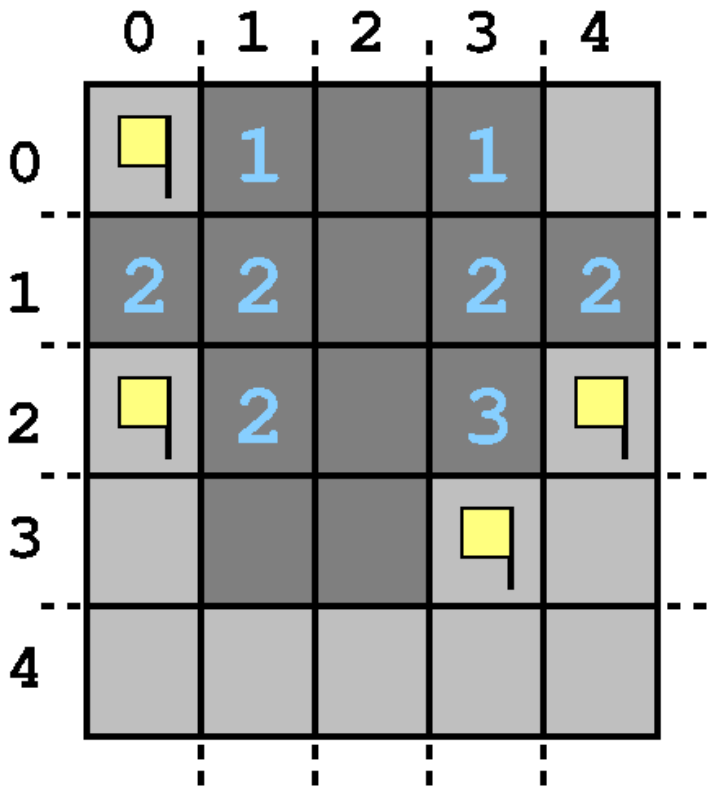
Case Study: Matrix Multiplication



```
type Matrix is { int[] data, int width, int height }
where |data| == (width * height)

function mul(Matrix A, Matrix B) -> (Matrix C)
// Arrays to multiply must be compatible
requires A.width == B.height
// Resulting array has specific dimension
ensures (C.width == A.width) && (C.height == b.height):
    ...
```

Case Study: Minesweeper




```
type ExposedSquare is {
  nat rank,
  bool holdsBomb
} where rank <= 8

type HiddenSquare is {
  bool holdsBomb,
  bool flagged
}

type Square is ExposedSquare|HiddenSquare

type Board is {
   Square[] squares,
   nat width,
   nat height
} where |squares| == (width * height)
```

http://whiley.org

@WhileyDave
http://github.com/Whiley