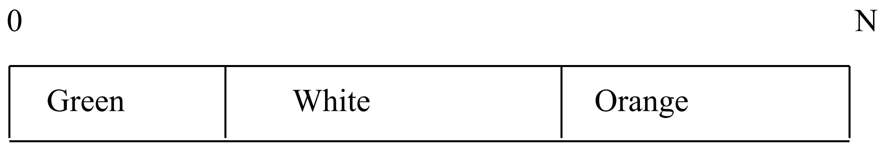
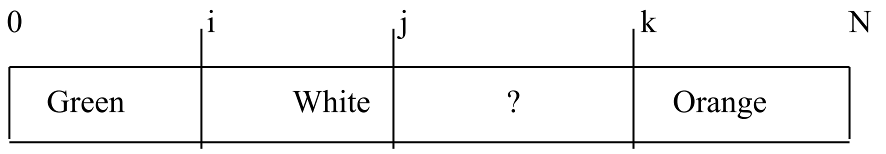
Chapter. The Irish National Flag.

In which we introduce the do..od statement informally.

Suppose we are given an array f[0..N) of coloured table tennis balls where {0 ≤ N}. The colours are green, white and orange. To start with the balls are arranged in some random order and what we have to do is to arrange them so that the array looks like this when we are finished. I am going to call this picture {Q}



Suppose we leave someone to carry out this task and we return at some stage during their work and observe the following situation.



How would we describe this picture in words? Well lets try….

“From the start up to but not including position i are green, from position i up to but not including position j are white, and from the end down to and including position k are orange”

I will want to use that sentence again but I really do not want to have to type it each time so I will let P stand for the sentence.

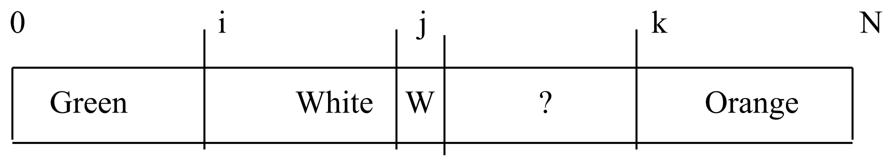
How would we measure the amount of work that is still to be done before we finish? Well a reasonable measure would be the size of the partition of the array labelled “?”. How big is that?

$$k - j$$

It is reasonable to suppose that as long as k – j is not zero then we still have work to do because we will still have a region labelled “?”.

Let us observe the next move? Suppose the person looking at f.j, now it can only contain a Green, a White or an Orange ball. Look at these in turn

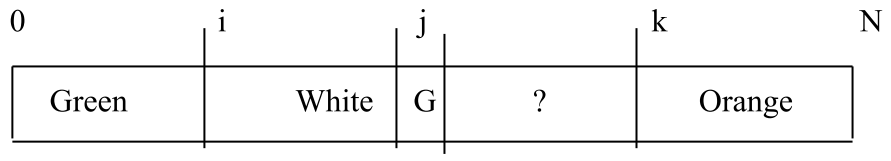
Case f.j = W



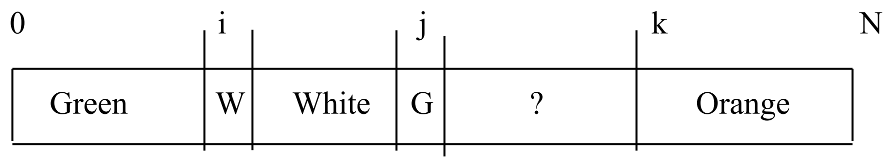
The most reasonable move to make here is to move j on by 1. This re-establishes P and reduces the value of k – j. So we write this as

If f.j = W → j := j + 1

Case f.j = G



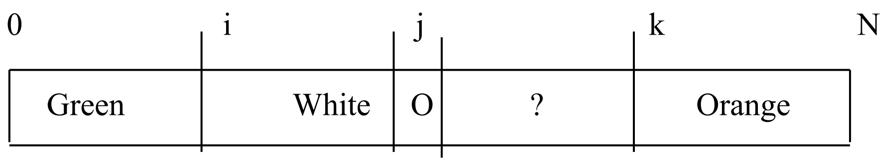
What should we do here? Well observe that f.i = W also



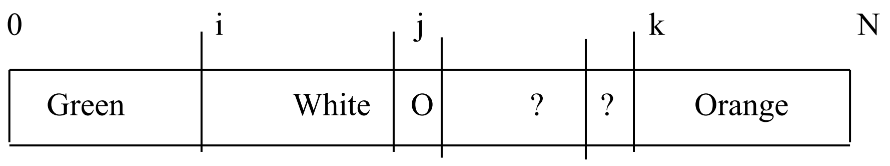
Probably the best thing to do here is to swap f.i with f.j and increase both i and j by 1. This re-establishes {P} and also reduce the value of k – j. So we write this as

If f.j = G → swap(f.i, f.j); i := i + 1, j := j + 1

Case f.j = O



What should we do here? Well observe that f.(k – 1) = ? also



The thing to do here is to swap f.j with f.(k – 1) and to reduce k by 1. This has the effect of re-establishing P and reducing k – j. We can write this as

```
If f.j = O →    swap(f.j, f.(k – 1)); k := k – 1
```

We have considered all 3 cases. I will write the set of possible moves as follows

```
If f.j = W  →   j := j + 1
[] f.j = G  →   swap(f.i, f.j); i := i + 1, j := j + 1
[] f.j = O  →   swap(f.j, f.(k – 1)), k := k – 1
fi
```

Once again, I want to refer to this in future so I will just call it S.

Now what have we learned?
I shall refer to the expression $k - j$ as vf (you can think of it as a measure of the amount of work left to do)

I shall refer to $j \neq k$ as B.

What we have is that

$$\{P \wedge B\}\ S\ \{P\}$$

If we start with the situation described by {P} and there still is an unexamined region, then doing S will re-establish {P}. But of course it does more than that, it reduced the unexplored region; it reduces vf. So suppose that before we perform S the value of vf is some value VF, then after we perform S the value of vf will be smaller. We could write this as

$$\{P \wedge B \wedge vf = VF\}\ S\ \{P \wedge vf < VF\}$$

When we have not finished we still have work to do can be written as

$$P \wedge B \Rightarrow 0 < vf$$

And when we have finished we will have achieved our goal

$$P \wedge \neg B \Rightarrow Q$$

I am using Q here to represent the very first picture in this little article.
All that remains is to ensure that {P} holds at the start. We can do so with the following assignment. You should satisfy yourself that this works.

$$i, j, k := 0, 0, N$$

Now if we put it all together we have constructed the following loop

```
i, j, k := 0, 0, N
;do j ≠ k →    {P ∧ B ∧ vf = VF}
               If f.j = W  →  j := j + 1
               [] f.j = G  →  swap(f.i, f.j); i := i + 1, j := j + 1
               [] f.j = O  →  swap(f.i, f.(k – 1)), k := k – 1
               Fi
               {P ∧  vf < VF}
od
{P ∧ ¬B} ⇒ {Q}
```

P is of course the “Loop Invariant”. B is the Loop Guard and vf is the variant.