```
modulo ConjAcotadoArr<T> implementa ConjAcotado<T>{
      var datos: Array<T>
      var largo: int
      Invariante de representacion para sin repetidos
      pred invRep (c: ConjAcotadoArr<T>) {
           (\forall i : \mathbb{Z}) \ (0 \le i \le c.largo \land_L \#(c.datos[i], subseq(c.datos, 0, c.largo) = 1)) \land 0 \le c.largo < c.datos.length
      }
      Invariante de representacion para con repetidos
      pred invRep (c: ConjAcotadoArr<T>) {
           0 \le c.largo \le c.datos.length
      Abstraccion
      pred Abst (c': ConjAcotadoArr<T>, c: ConjAcotado<T>) {
           c.cap = c'.datos.length \land (\forall e : T) \ (e \in c.elems \leftrightarrow (\exists p : \mathbb{Z}) \ (0 \le p < c'.largo \land_L c'.datos[p] = e))
      algoritmos
      impl agregar(inout c': ConjAcotadoArr<T>, in e: T):{
             if c'.pertenece(e) then
                return (para terminar el programa basicamente)
             else
                c.datos[c.largo]:= e
                c'.largo:= c'.largo+1
              end if
             return
      impl sacar(inout c: ConjAcotadoArr<T>, in e: T):{
             if !c.pertenece(e) then
                skip
             else
                int i:= 0
                   while c.elems[i] \neq e do
                       i++
                   end while
                   c.elems[i]:= c.elems[c.largo]
              end if
             c.largo:= c.largo-1
          }
      impl pertenece(inout c: ConjAcotadoArr<T>, in e: T):Bool{
             int i:= 0
             while i < c.elems.length do
               if c.elems[i] = e then
                  return true
               else
                  skip
                end if
               i++
               end while
             return false
      while hola do
         hola2
         while hi do
             things
          end while
      end while
```

```
\quad \textbf{if} \ \mathsf{cosa} \ \textbf{then} \\
        hi
    else
        bie
      end if
    while cosa do
        hola
        qui:= sou
        \quad \textbf{if} \ \text{fiu then} \\
             fu
         else
             fa
          end if
    end while
    var hola|: queso
}
```

¿Cual es mas eficiente, cuanto usarian una u otra?

En cuento a velocidad probablemente sea mas eficiente con repetidos, pero en cuanto a capacidad yo creo que sin repetidos es mas eficiente