```
prepare input artefacts()

finished ≔ false
readOnlyBlackboardView ≔ Read-Only Blackboard View .construct (blackboard)
measurementControllerBlackboardView ≔ Measurement Controller Blackboard View .construct (blackboard)
analyserBlackboardView ≔ Analyser Blackboard View .construct (blackboard)
finalJudgeBlackboardView ≔ Final Judge Blackboard View .construct (blackboard)

while (¬finished) do
    if Measurement Controller .can measure (readOnlyBlackboardView) then
   Measurement Controller .measure (measurementControllerBlackboardView)
    else
        analysersIterator ≔ Analysers .iterator()
        while ¬analysersIterator.current().can Contribute (readOnlyBlackboardView) do
            analysersIterator.next()
        od
        if analysersIterator.current().can Contribute (readOnlyBlackboardView)
            analysersIterator.current().contribute (analyserBlackboardView)
        else
            finished ≔ Final Judge .judge(finalJudgeBlackboardView)
        fi
    fi
od

write results()
```