« Le second [précept], de diviser chacune des difficultés que j'examinerais, en autant de parcelles qu'il se pourrait et qu'il serait requis pour les mieux résoudre. »

René Descartes, Discours de la méthode, seconde partie, 1637.

```
Algorithme NombrePremierVersion1
(* Demande un nombre positif, jusqu'à ce que l'utilisateur donne un nombre premier *)
var nombre:entier      (* le nombre demandé *)
    premier:booléen    (* vrai ssi nombre est premier *)
    estdivpari:booléen (* vrai ssi nombre est divisible par i *)
    i:entier           (* compteur de boucle pour le test des diviseurs *)
    p:entier           (* partie entière de la racine carrée de nombre *)

début
   écrire("Pour trouver un nombre premier ...")
   répéter
       écrire("Donner un entier strictement positif : ") (* début lire positif *)
       lire(nombre)
       tant que nombre ≤ 0 faire
            écrire("Donner un entier, qui doit être strictement positif : ")
            lire(nombre)
       fin tant que                                      (* fin lire positif *)

       si nombre = 1 alors                               (* début tester si premier *)
            premier ← faux
       sinon
            premier ← vrai

            p ← 1                                             (* début calcul racine *)
            tant que (p+1)×(p+1) ≤ nombre faire
                 p ← p + 1
            fin tant que                                      (* fin calcul racine *)

            pour i de 2 à p faire
                 estdivpari ← (nombre mod i) = 0              (* début test divisible *)
                 si estdivpari alors
                      écrire(nombre, " est divisible par ", i, " et par ", nombre div i)
                 fin si                                       (* fin test divisible *)

                 premier ← premier et non estdivpari
            fin pour
       fin si                                            (* fin tester si premier *)
   jusqu'à premier
   écrire("Bravo, vous avez trouvé un nombre premier : ", nombre)
fin
```

```
Algorithme NombrePremierVersion2
(* Demande un nombre positif, jusqu'à ce que l'utilisateur donne un nombre premier *)
var nombre:entier      (* le nombre demandé *)
    premier:booléen    (* vrai ssi nombre est premier *)

    fonction lire_positif:entier
    (* lit un entier jusqu'à ce qu'il soit positif *)
    var n:entier
    début
        écrire("entrer un nombre entier positif :")
        lire(n)
        tant que n ≤ 0 faire
            écrire("positif svp !")
            lire(n)
        fin tant que
        retour(n)
    fin

    fonction teste_premier(n:entier):booléen
    (* n est premier ? affiche diviseurs le cas échéant *)
    var i:entier
        premier:booléen

          fonction racine(n:entier):entier
          (* calcule la partie entière de la racine de n (n>0) *)
          var p:entier
          début
              p ← 1
              tant que (p+1)×(p+1) ≤ n faire
                   p ← p + 1
              fin tant que
              retour(p)
          fin

          fonction divisible_par_i(n,i:entier):booléen
          (* n divisible par i ? si oui affiche diviseurs *)
          début
              si (n mod i) = 0 alors
                   écrire(n, " est divisible par ", i, " et par ", n div i)
                   retour(vrai)
              sinon
                   retour(faux)
              fin si
          fin

    début
        si n=1 alors retour(faux)
        sinon
            premier ← vrai
            pour i de 2 à racine(n) faire
                premier ← premier et non divisible_par_i(n,i)
            fin pour
            retour(premier)
        fin si
    fin

début
    premier ← faux

    tant que non premier faire
         nombre ← lire_positif
         premier ← teste_premier(nombre)
    fin tant que

    écrire("bravo ", nombre, " est premier")
fin
```