Les itérations

Un premier algorithme

Algorithme Carré
Variables unNombre, sonCarré : entiers {déclarations : Réservation d'espace mémoire}

début

{préparation du traitement}

ecrire("Quel nombre voulez-vous élever au carré?")
lire(unNombre)

{traitement : calcul du carré}

SonCarré ← unNombre×unNombre

{présentation du résultat}

ecrire("Le carré de ", unNombre, "c'est ", sonCarré)

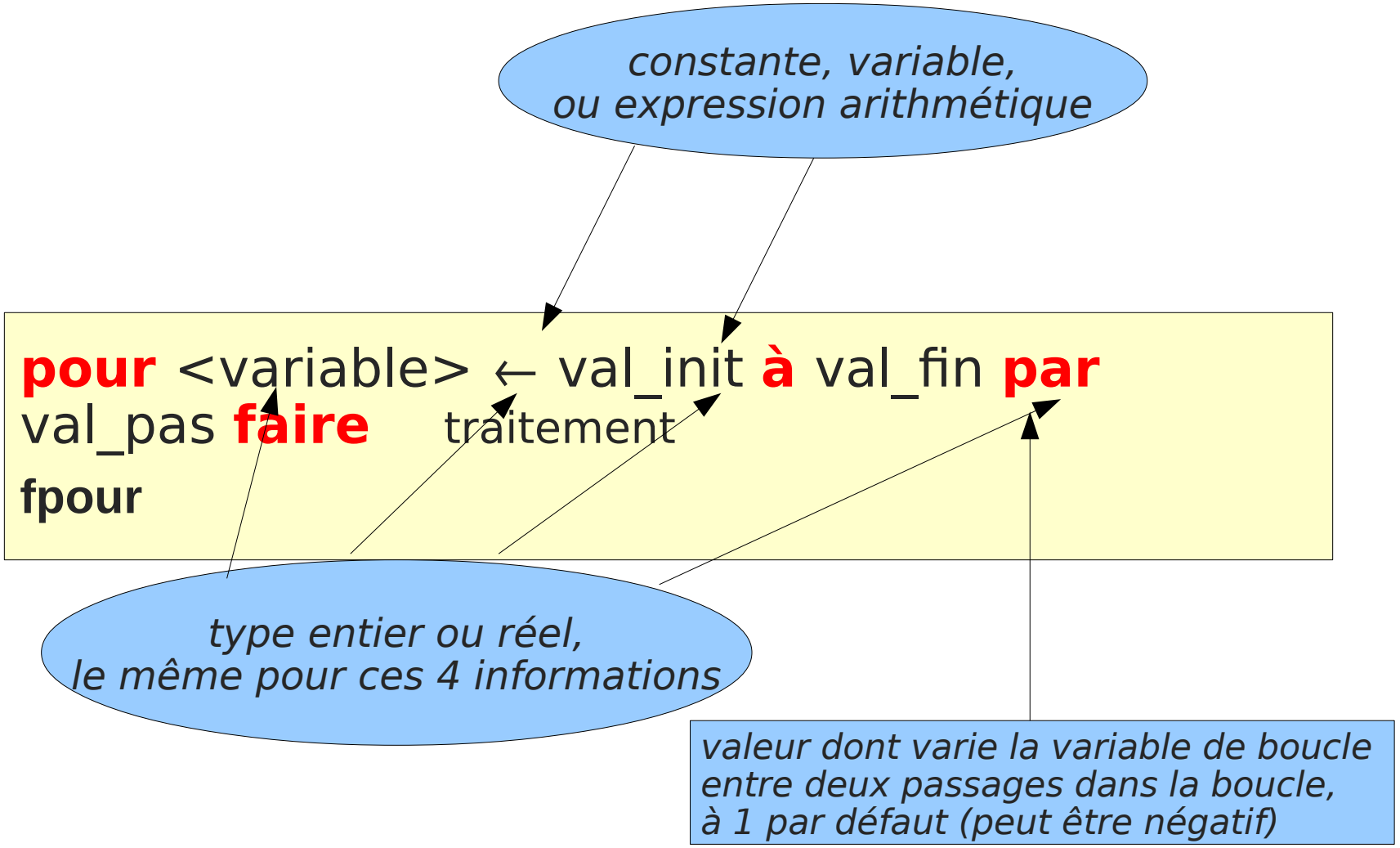
fin

La boucle « pour »

pour <var> ← val_init **à** val_fin [**par** <pas>] **faire** traitement
{suite d'instructions}
fpour

- **Fonction** :
  - répéter une suite d'instructions un certain nombre de fois

Les champs de la boucle « pour »



Sémantique de la boucle pour

- **Implicitement, l'instruction pour** :
  - initialise une variable de boucle (le compteur)
  - incrémente cette variable à chaque pas
  - vérifie que cette variable ne dépasse pas la borne supérieure

- Attention :
  - le traitement ne doit pas modifier la variable de boucle

pour i ←1 à MAX **faire**

si (…) **alors i** ←MAX

fpour

Quand le nombre d'itérations n’est pas connu…

```
Algorithme somme
// fait la somme des données qu’il saisit, arrêt à la lecture d'une valeur <= 0
variables val, somme : entiers
début
somme ← 0
Écrire("Donnez une valeur")                          {amorçage}
Lire(val)
tant que (val > 0) faire
        somme ← somme + val                          {traitement}
        Écrire("Donnez une autre valeur")
        Lire(val)                                    {relance}
ftq
Écrire("La somme des valeurs saisies est " , somme)
fin
```

La boucle « tant que .. faire»

amorçage *{initialisation de la (des) variable(s) de condition}*

Tant que <expression logique (vraie)> **faire**

traitement *{suite d'instructions}*

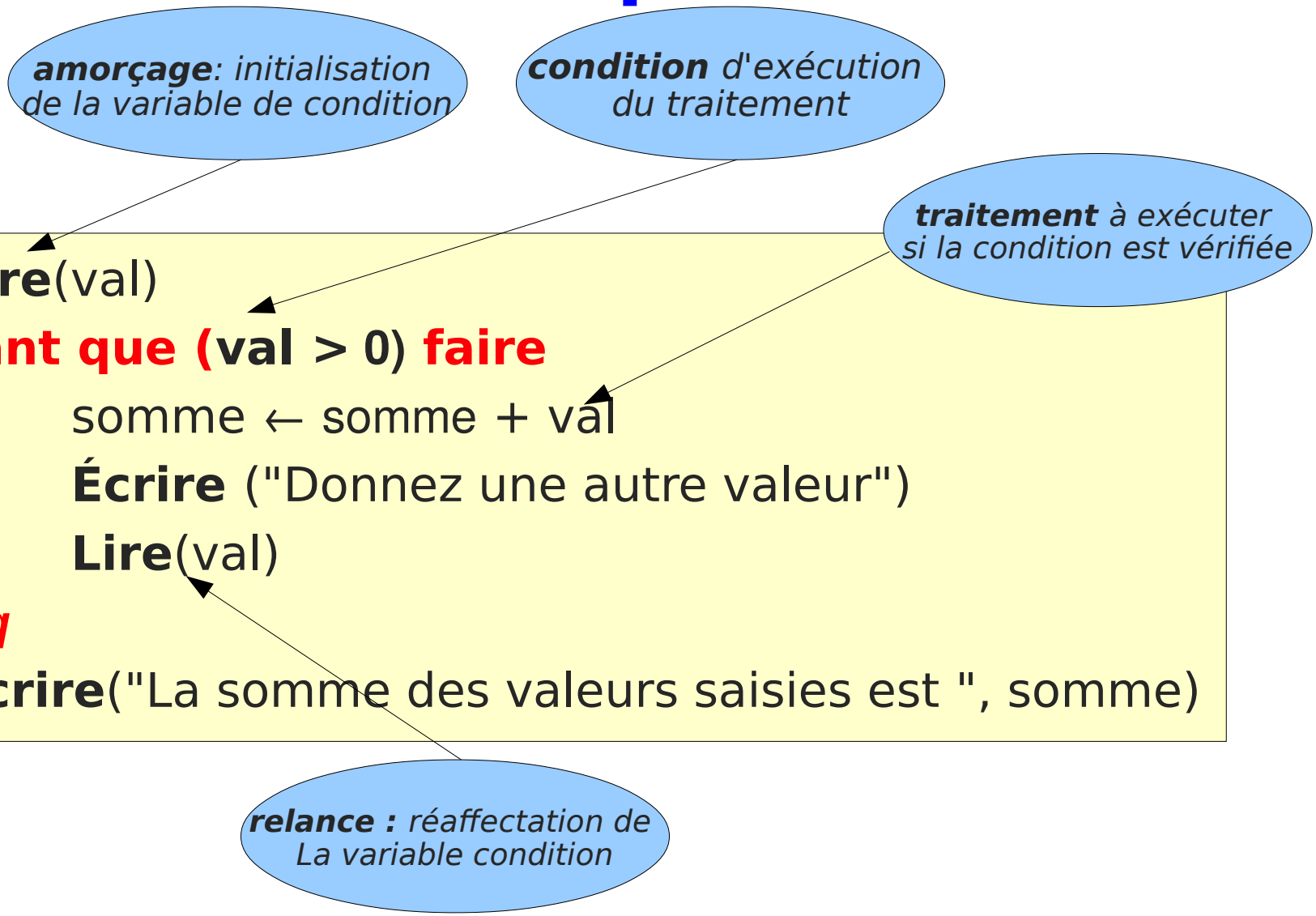
relance *{ré-affectation de la (des) variable(s) de condition}*

ftq

- Fonction: répéter une suite d'instructions tant qu'une condition est remplie

- Remarque : si la condition est fausse dès le départ, le traitement n'est **jamais** exécuté

Sémantique de la boucle tant que



Comparaison boucles « pour » et « tant que »

```
pour i ← 1 à nbVal faire
        Écrire("Donnez une valeur :")
        Lire(valeur)
        sommet ← somme + valeur
fpour
```

… équivaut à :

```
i ← 0
Tant que (i < nbVal) faire
        Écrire("Donnez une valeur :")
        Lire(valeur)
        sommet ← somme + valeur
        i ← i + 1
ftq
```

Choisir pour… Choir tant que

- si le nombre d’itérations est connu à l’avance,
  ↪**choisir la boucle pour**

- si la boucle doit s'arrêter quand survient un évènement,
  ↪**choisir la boucle tant que**

La boucle répéter : un exemple

```
Algorithme Essai
// saisit d'une valeur positive paire, répéter tant que la valeur est
// négative
variables valeur : entier
Début
répéter
        Écrire("Donnez une valeur positive non nulle : ")
        Lire(valeur)
tant que (valeur < 0 ou valeur % 2 <> 0)

Écrire("La valeur positive piare que vous avez saisie est ")
Écrire(valeur )
... {traitement de la valeur saisie}
fin
```

La boucle «répéter ...tant que»

répéter

(ré)affectation de la (des) variable(s) de condition

traitement *{suite d'instructions}*

tant que <expression logique (vraie)>

- **Fonction**:
  - exécuter une suite d'instructions ***au moins une fois*** et la répéter tant qu'une condition est remplie

- **Remarque**: le traitement dans l'exemple précédent se limite à la réaffectation de la variable de condition

Comparaison boucles « répéter » et « tant que »

```
répéter
        Écrire("Donnez une valeur positive paire :")
        Lire(valeur)
Tant que (valeur < 0 ou valeur % 2 <> 0)
```

… équivaut à :

```
Écrire("Donnez une valeur positive paire :")
Lire(valeur)
Tant que ( (valeur < 0) ou (valeur % 2 <> 0) ) faire
        Écrire("Donnez une valeur positive paire :")
        Lire(valeur)
ftq
```

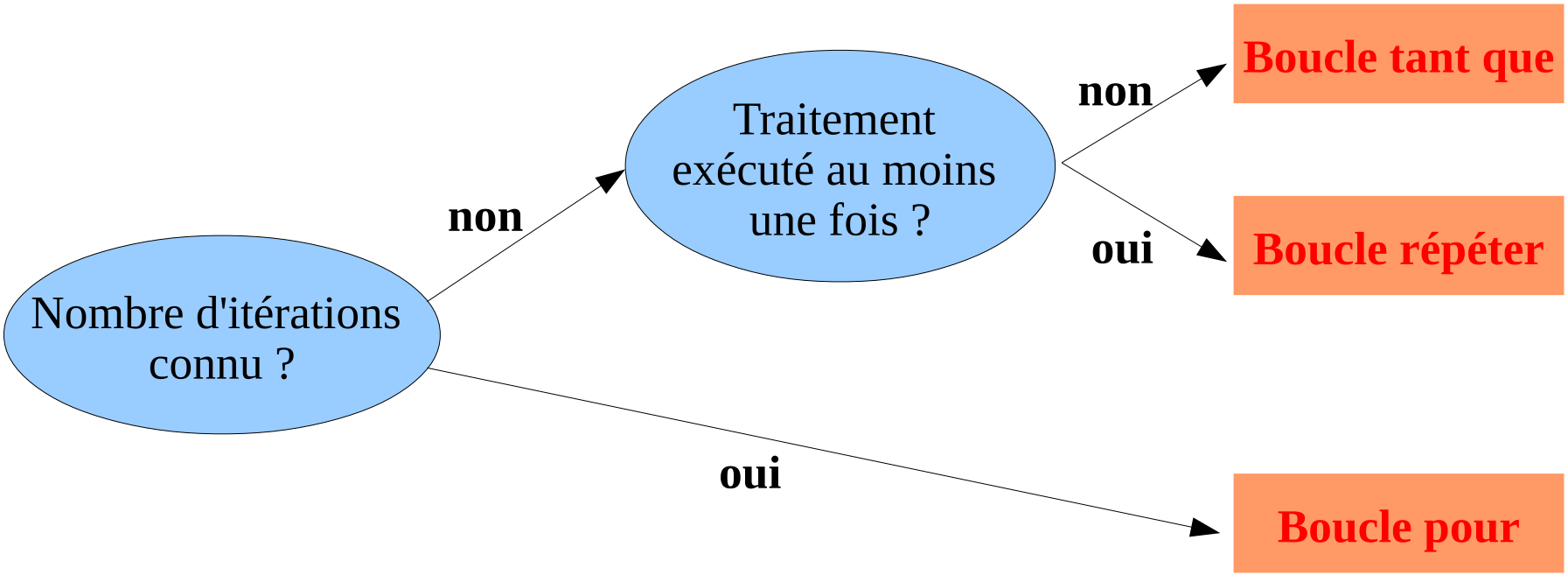
Comparaison boucles « répéter » et « tant que »

- **Boucle «tant que »**
  - condition vérifiée **avant** chaque exécution du traitement
  - le traitement peut donc ne pas être exécuté
  - de plus : la condition porte surtout sur la saisie de nouvelles variables (relance)

- **Boucle « répéter tant que »**
  - condition vérifiée après chaque exécution du traitement
  - le traitement est exécuté au moins une fois
  - de plus : la condition porte surtout sur le résultat du traitement

Remarque : la boucle répéter est typique pour les saisies avec vérification.

Choisir pour… tant que... répéter...



Le problème d'une boucle : il faut en sortir !

tant que A faire B

répéter B tant que A

- La suite d'instructions B doit amener A à prendre la valeur Faux.
  - La suite d'instructions B doit modifier au moins une variable de l'expression logique A
  - Exemple (à ne pas faire) :

```
X ← 1 ; Y ← 1
Tant que ( X <= 999) faire
    Y ← X*X
    Écrire ("Le carré de ", X, "est ",Y)
Fintq
```

Remarque : C'est l'expression logique A (et elle seule!) qui en prenant la valeur Faux provoque l'arrêt de la boucle.

Illustration des boucles

Afficher le carré des valeurs saisies **tant que qu'on ne saisit pas 0**

```
Lire (val)
Tant que (val <> 0) faire
        Écrire (val * val)
        Lire (val)
Fintq
```

Saisir des valeurs et s'arrêter **dès que leur somme dépasse 1000**

```
Écrire("Donnez une valeur:")
Lire(val)
Somme ← val
Tant que ( somme <= 1000) faire
    Écrire("Donnez une valeur:")
    Lire(val)
     Somme ← somme + val
ftq
```

Illustration des boucles (suite)

Saisir des valeurs et s'arrêter **dès que leur somme dépasse 1000**

```
Somme ← 0
répéter
    Écrire("Donnez une valeur:")
    Lire(val)
     Somme ← somme + val
Tant que (somme <= 1000)
```

Remarque : si on veut afficher le résultat final, il faut soustraire la dernière valeur saisie (val) de Somme à la sortie de la boucle.

Exemple d'un mauvais choix de boucle

```
Algorithme Somme
// fait la somme d'une suite de valeurs tant que cette somme
// ne dépasse pas un seuil donné.
Constante (seuil : entier) ← 1000
variables val, Somme : entier
Début
Somme ← 0
répéter
        Écrire("Donnez une valeur : ")
        Lire(valeur)
        Somme  ← Somme + val
tant que (Somme ≤ seuil)

Écrire("La somme finale est ", Somme - val)
fin
```

Quand utiliser la boucle tant que

- Dès que la condition d'itération devient très complexe, il est conseillé d'utiliser la boucle tant que.
  - Exemple:

    saisir des valeurs, les traiter, et s'arrêter à la saisie de la valeur d'arrêt **–1** ou après avoir saisi **5** données.

```
Algorithme Essai
Constantes (STOP: entier) ← -1
                  (MAX : entier) ← 5
variables cpt, valeur : entier
Début
cpt ← 0
Lire (valeur)
Tant que ( (valeur ≠ STOP) et (cpt < MAX) ) faire
          Cpt++
        //traitement de la valeur saisie et relance
        ...
          Lire(valeur)
Fintq
Écrire(valeur,cpt) // {traitement de la valeur saisie}
fin
```

Interpréter l'arrêt des itérations

```
Algorithme Essai
Constantes (STOP: entier) ← -1
                (MAX : entier) ← 5
variables cpt, valeur : entier
Début
cpt ← 0
Lire (valeur)
Tant que ( (valeur ≠ STOP) et (cpt < MAX) ) faire
    cpt ← cpt + 1
      … // {traitement de la valeur saisie et relance}
    Lire(valeur)
Fintq
Si (valeur == STOP)
    alors {la dernière valeur testée était la valeur d'arrêt}
            Écrire("Sortie de boucle car saisie de la valeur d'arrêt")
Sinon {il y avait plus de 5 valeurs à tester}
            Écrire("Sortie de boucle car nombre maximum de valeurs
             à traiter atteint ")
fin
```

Importance du test de sortie de boucle

Tant que (valeur ≠ STOP) ET (cpt < MAX) **faire**

- **Dans la boucle: val ≠ STOP et nbVal < MAX** est vrai

- **A la sortie de boucle**:
  - Soit (val≠STOP) est faux → val=STOP
  - soit (nbVal < MAX) est faux → nbVal ≥ MAX

- Que tester à la sortie de boucle ?
  - **si (val == STOP) alors ...** voir transparent précédent.
  - **si (cpt ≥ MAX) alors ...** mauvais test car message dépend de la dernière valeur saisie.

Retour sur les conditions d'itération

- Interpréter (et bien comprendre!) l'arrêt des itérations à la sortie d'une boucle.

tant que <cond> **faire**

À la sortie :	**non**(<cond>) *est vrai*
donc si	cond = p **et** q
à la sortie:	**non** (p **et** q)
c'est à dire	**non** p **ou non** q

- Exemple : avec **<cond>** égal à : valeur ≠ STOP **et** cpt < MAX

  non(<cond>) égal à : val = STOP **ou** cpt ≥ MAX

Retour sur les conditions d'itération (suite)

- **Simplifier** une écriture par substitution d'une formule équivalente

  si (Age = "Mineur"

  ou (non (Age = "Mineur") et non (Fisc = "Imposable"))) alors...

Equivalent à :

si (Age = "Mineur" ou non (Fisc = "Imposable")) alors...

- **Vérifier la validité d'une condition**

  si valeur < 10 et Valeur > 100 alors... ***cas improbable***

- **Ecrire la négation d'une condition**

  si on veut P et Q et R :

  répéter .... tant que **non** P **ou non** Q **ou non** R **ou ..**