Exo 7 : Le jeu du juste prix.

La calculatrice détermine au hasard le juste prix

(un nombre caché **X** entier entre 0 et 1000),

et vous devez le trouver le plus rapidement possible.

A chaque tentative **Y**, elle vous informe s'il vous faut viser plus haut ou plus bas,

et à la fin vous informe en un nombre **N** de coups vous avez trouvé le nombre caché.

Le jeu du juste prix.

La calculatrice détermine au hasard le juste prix

(un nombre caché **X** entier entre 0 et 1000),

et vous devez le trouver le plus rapidement possible.

A chaque tentative **Y**, elle vous informe s'il vous faut viser plus haut ou plus bas,

et à la fin vous informe en combien de coups vous avez trouvé le nombre caché.

Etape 1 : l'organigramme est-il à actions successives ?

Le jeu du juste prix.

La calculatrice détermine au hasard le juste prix

(un nombre caché **X** entier entre 0 et 1000),

et vous devez le trouver le plus rapidement possible.

A chaque tentative **Y**, elle vous informe s'il vous faut viser plus haut ou plus bas,

et à la fin vous informe en combien de coups vous avez trouvé le nombre caché.

Etape 1 : l'organigramme est-il à actions successives ?

Non, car « Viser plus haut » (ou « plus bas ») ne sera pas forcément exécutée.

Non, car le nombre d'actions ne sera pas toujours le même, selon que l'on devine le nombre caché rapidement ou pas.

Etape 1 : l’organigramme est-il à condition ?

…

Etape 1 : l'organigramme est-il à condition ?

Oui, car la machine nous informera que soit il faut viser plus haut, soit il faut viser plus bas.

Etape 1 : l'organigramme est-il à condition ?

Oui, car la machine nous informera que soit il faut viser plus haut, soit il faut viser plus bas.

Peut-on, avec cette condition, réaliser un organigramme à actions successives ?

Etape 1 : l'organigramme est-il à condition ?

Oui, car la machine nous informera que soit il faut viser plus haut, soit il faut viser plus bas.

Peut-on, avec cette condition, réaliser un organigramme à actions successives ?

Non, car on ne sait pas combien en mettre (on ne sait pas en combien de coups on va gagner).

Comment placer un nombre de fois variable une même action ?

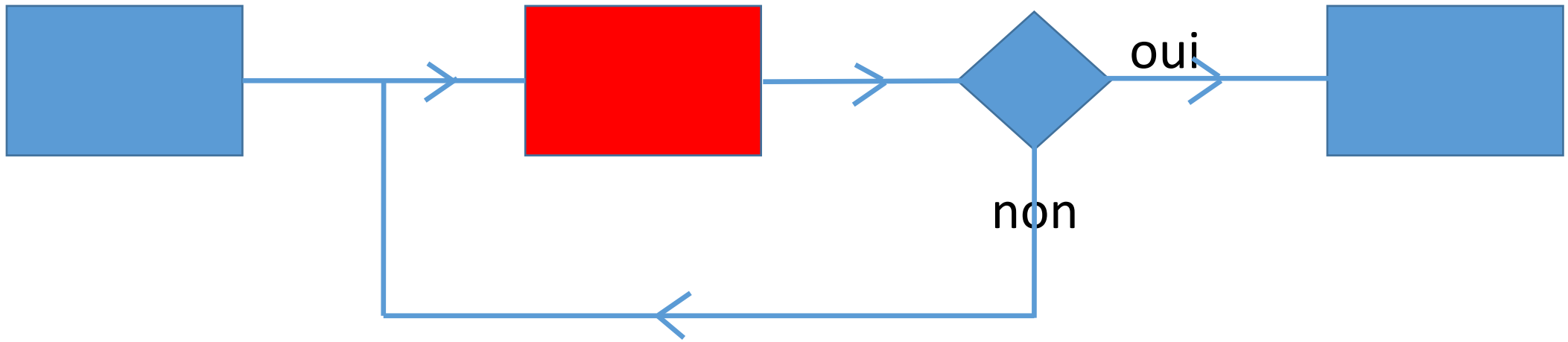
En mettant cette action dans …

Comment placer un nombre de fois variable une même action ?

En mettant cette action connue dans une boucle, et en mettant une condition permettant de repasser par cette action un nombre de fois variable.

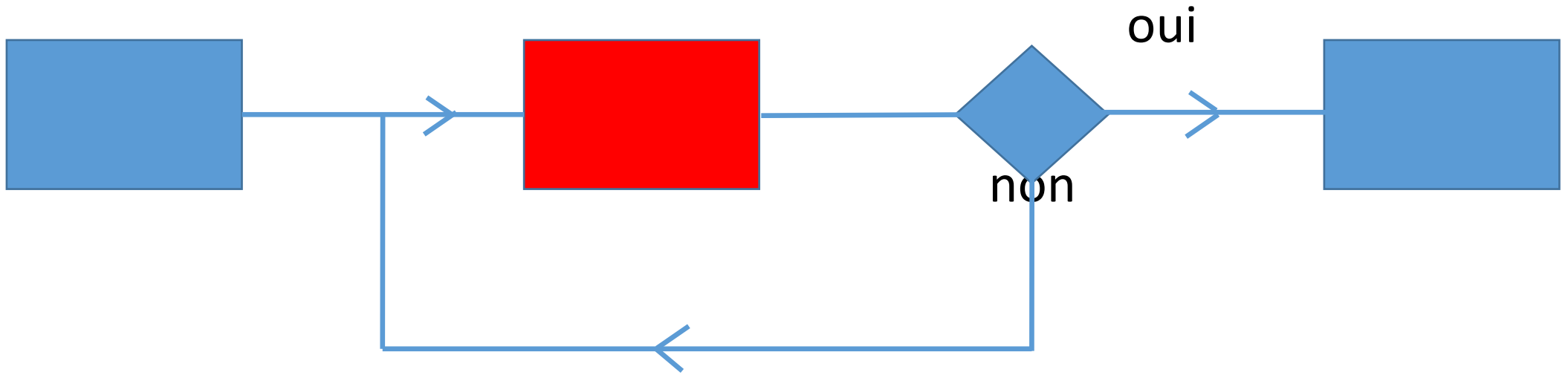
Comment placer un nombre de fois variable une même action ?

En mettant cette action dans une boucle, et en mettant une condition permettant de repasser par cette action un nombre de fois variable.



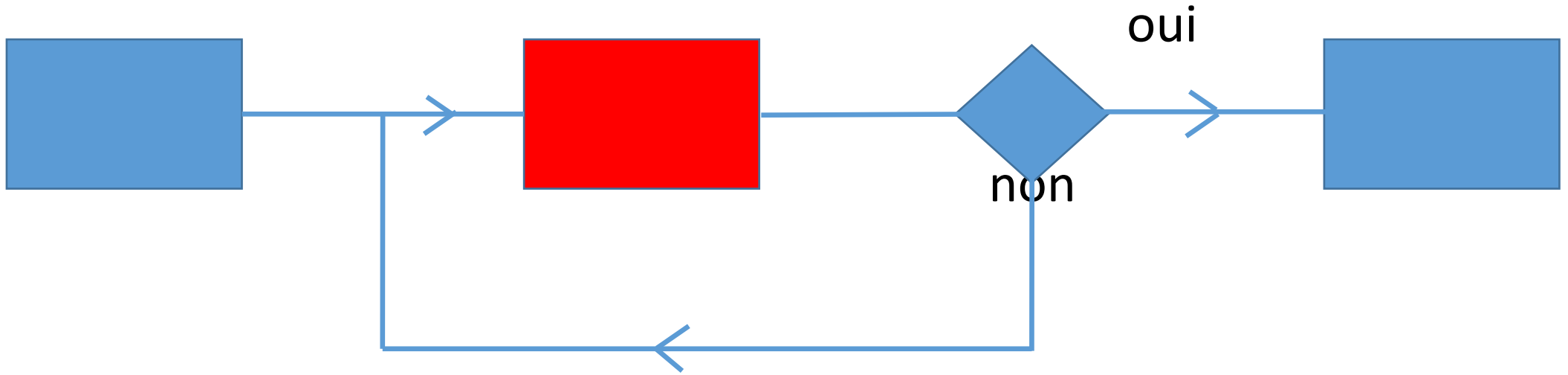
Cet organigramme est-il un organigramme à condition ?

Oui, car ...



Cet organigramme est-il un organigramme à condition ?

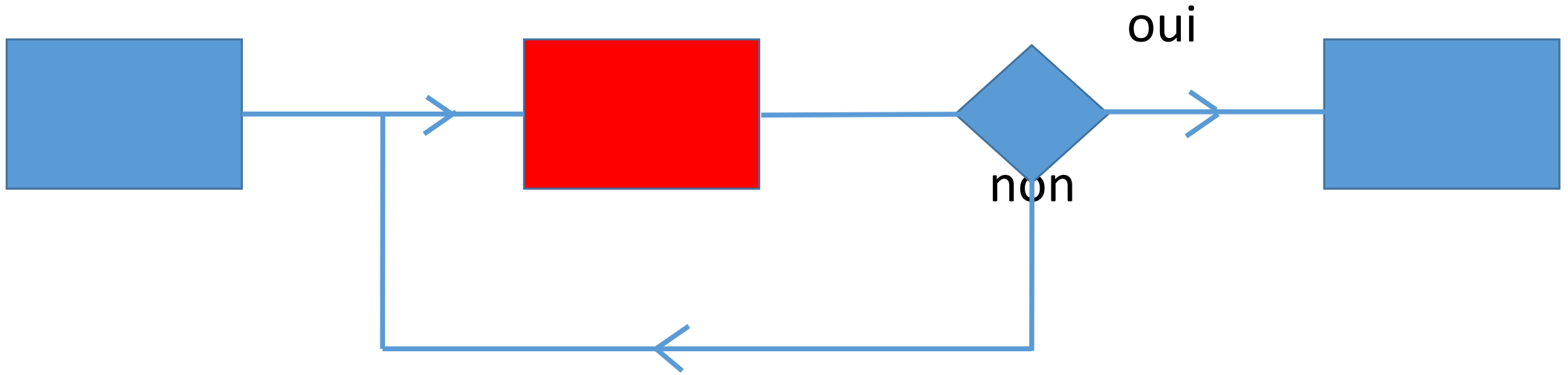
Oui, car il y a une condition.



Cet organigramme est-il un organigramme à condition ?

Oui, car il y a une condition.

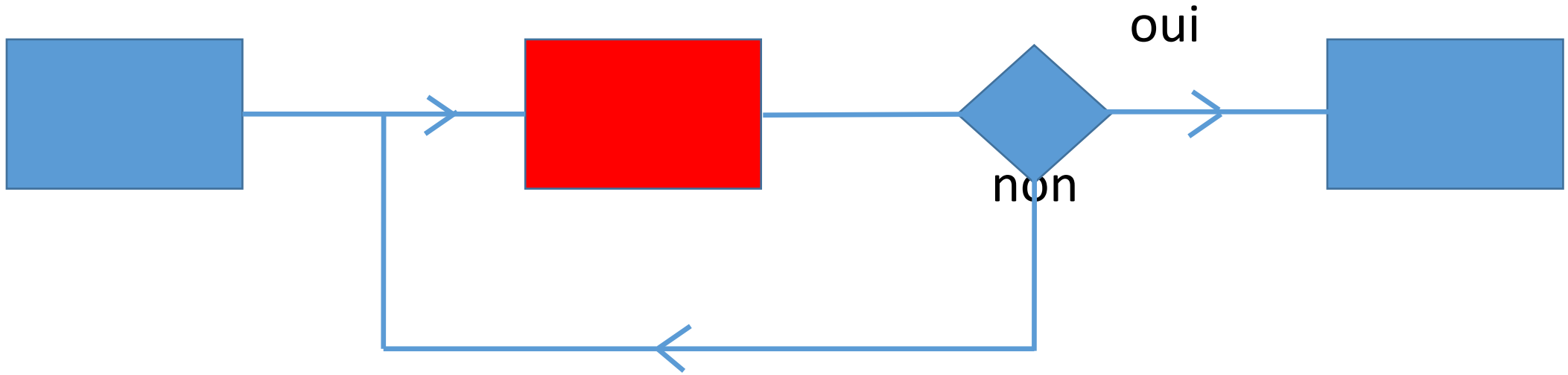
Non, car …



Cet organigramme est-il un organigramme à condition ?

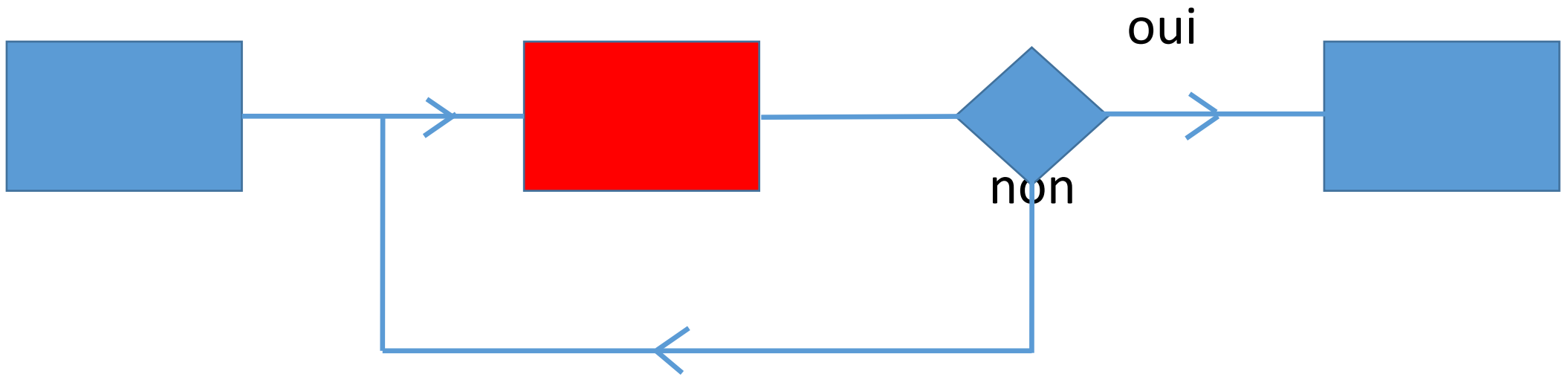
Oui, car il y a une condition.

Non, car ce n’est pas le cas soit une action, soit l’autre.



Quelle est la différence entre un organigramme à condition

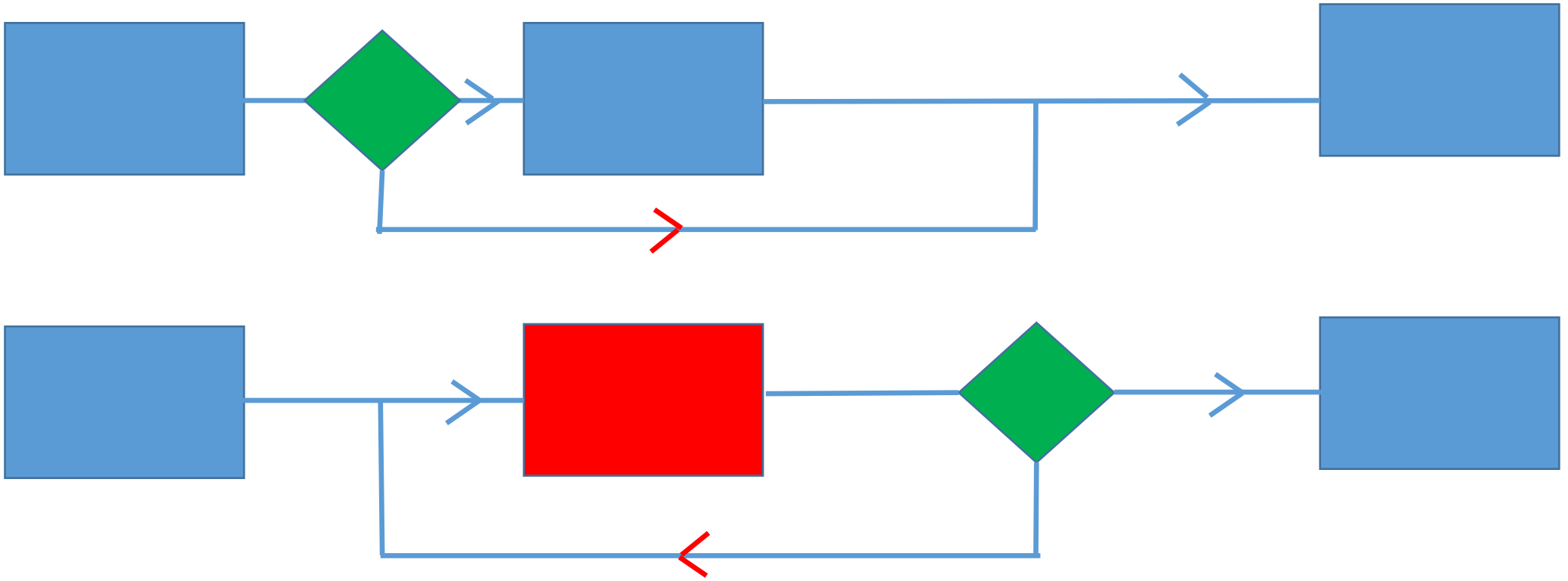
et un organigramme à boucle ?



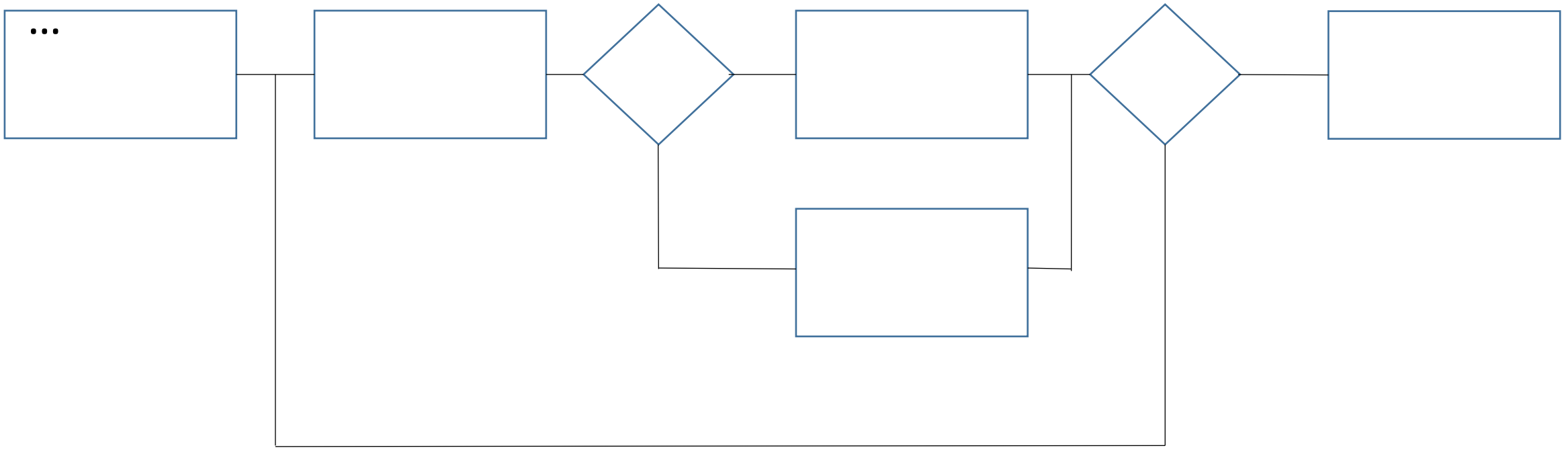
Quelle est la différence entre un organigramme à condition

et un organigramme à boucle ?

La condition est placé avant l'action pour un organigramme à condition, et après l'action pour un organigramme à boucle.



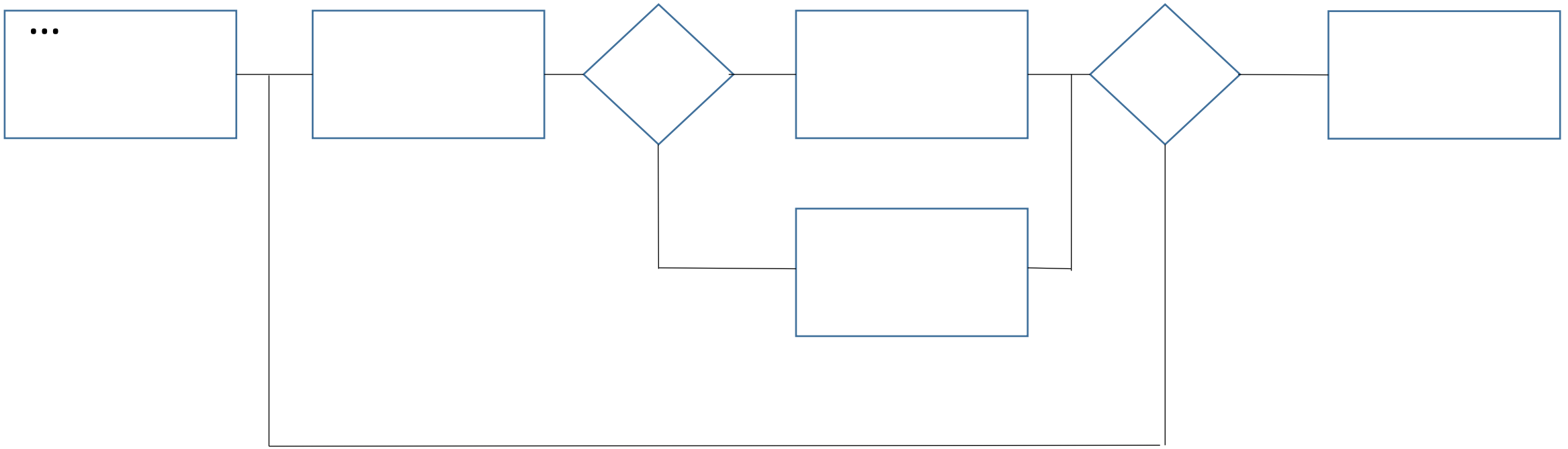
Organigramme :



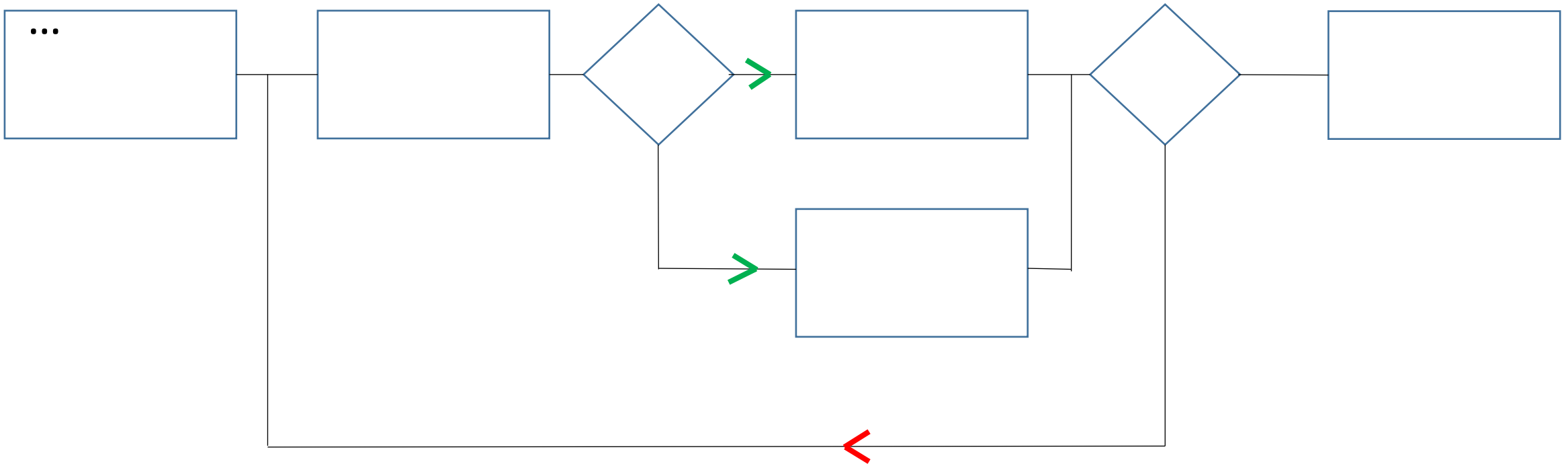
X = le nombre caché Y = mes tentatives

Elle doit m’informer si je dois viser plus haut ou plus bas.

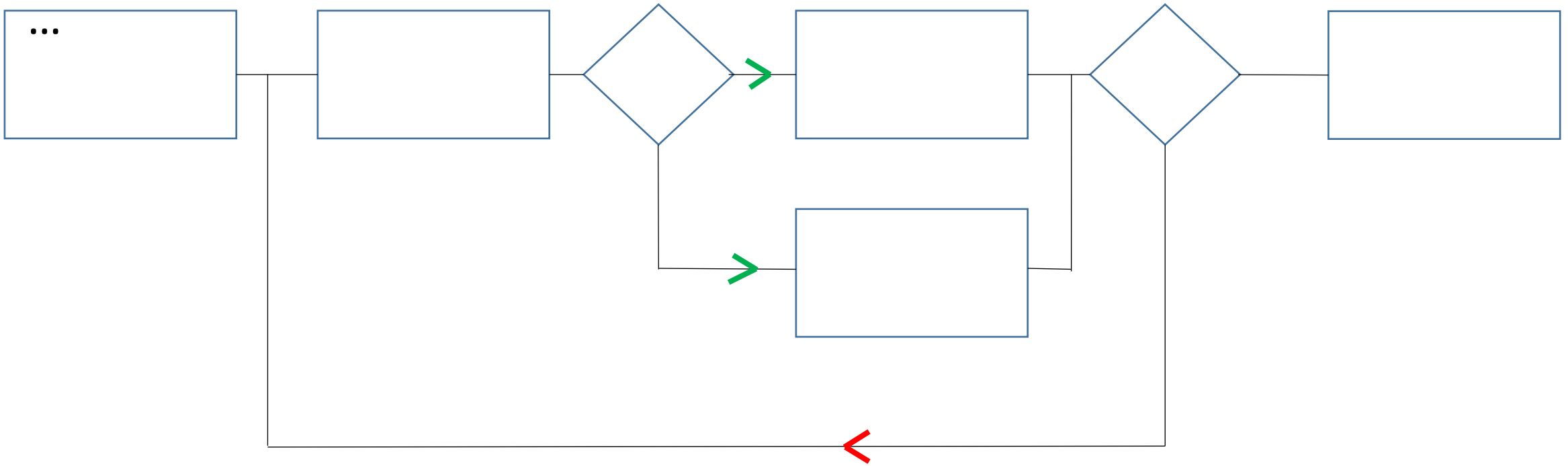
Organigramme : quels sont les sens de circulation ?



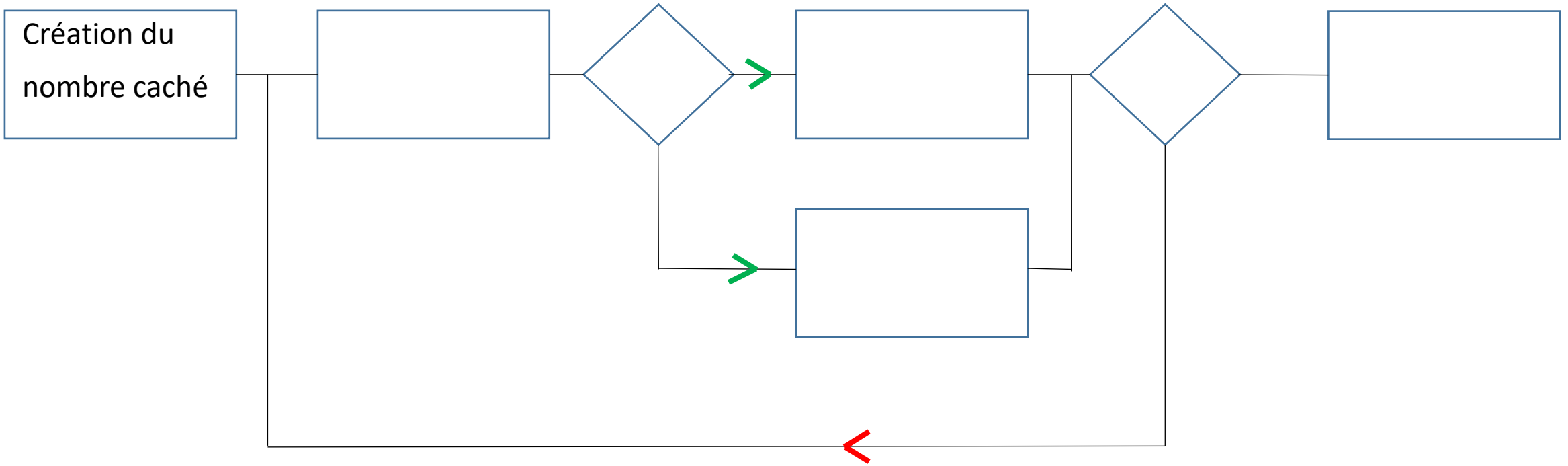
Organigramme : quels sont les sens de circulation ?



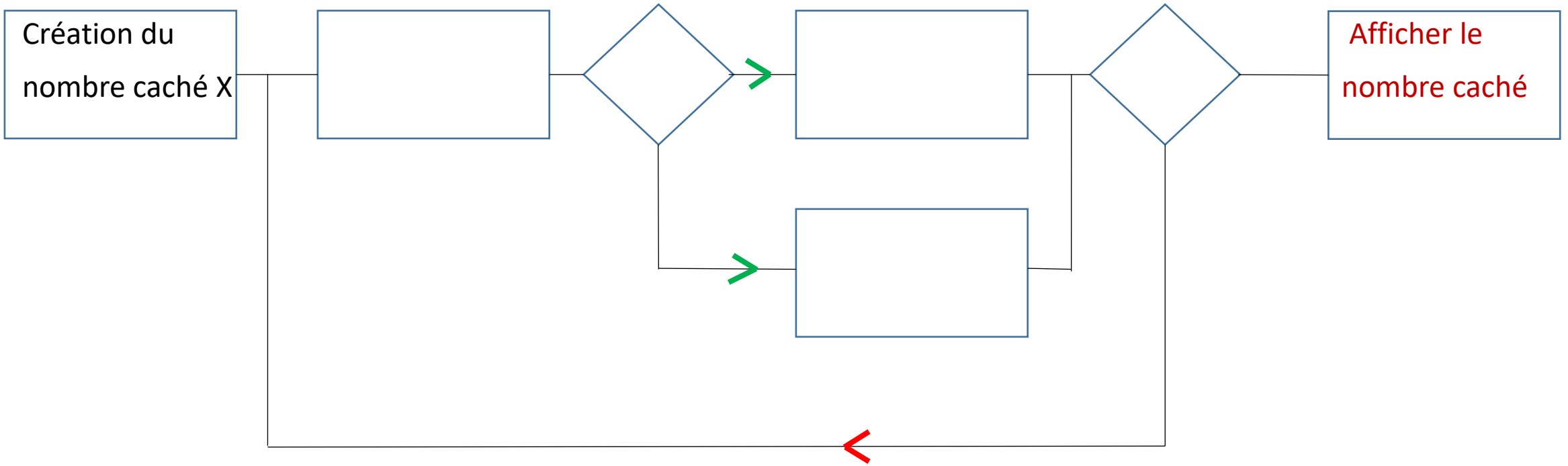
Première action ?



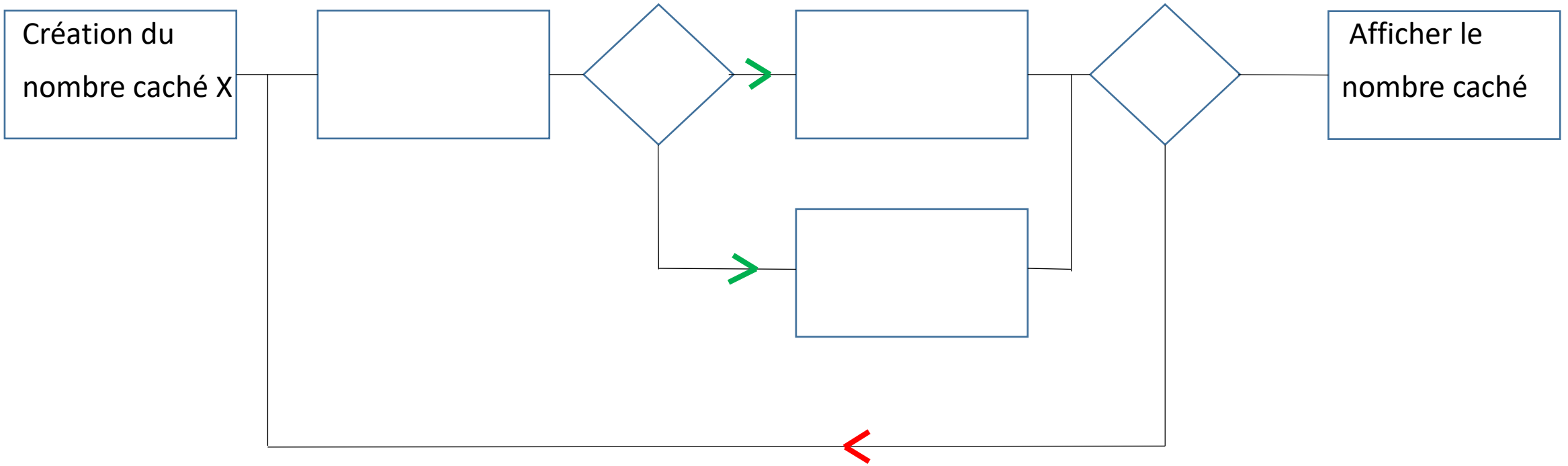
Première action ?



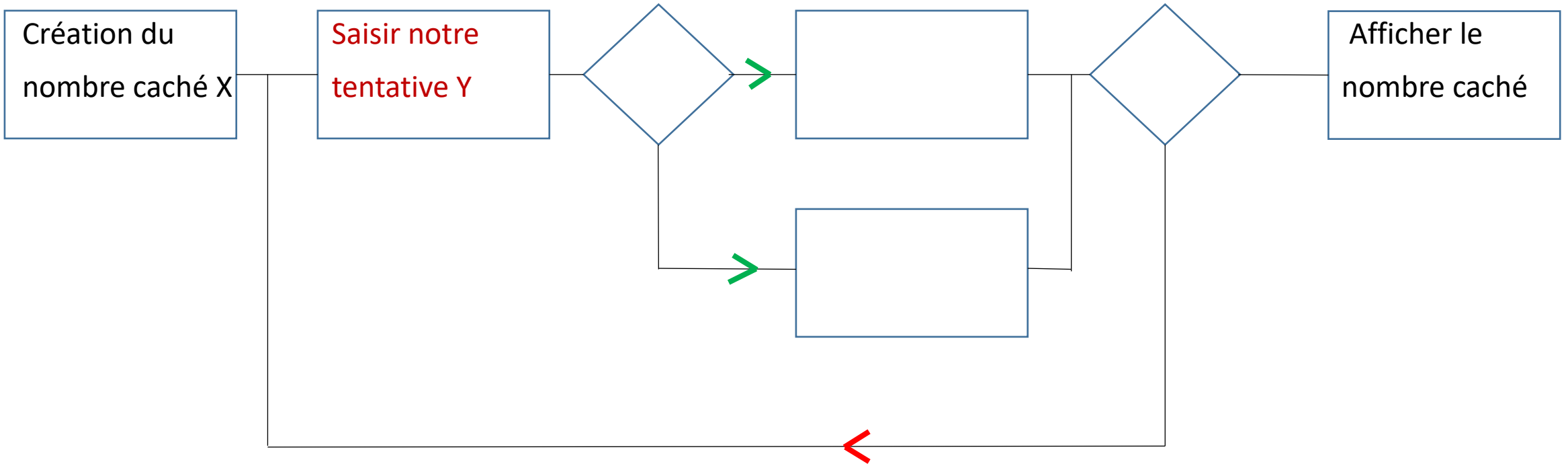
Dernière action ?



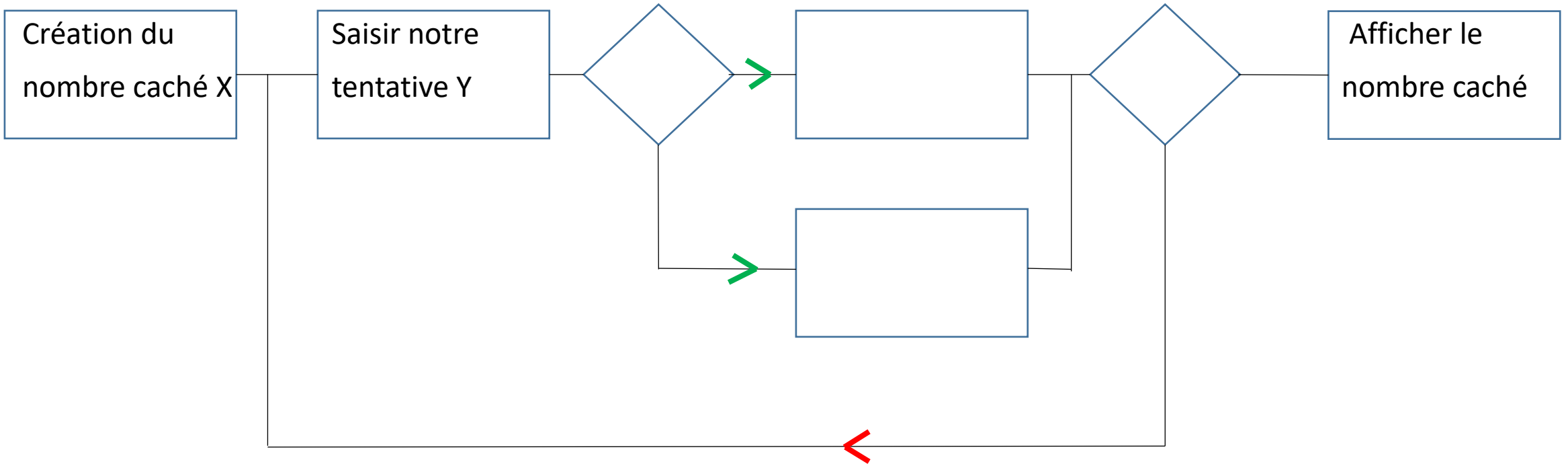
Où faire nos tentatives ?



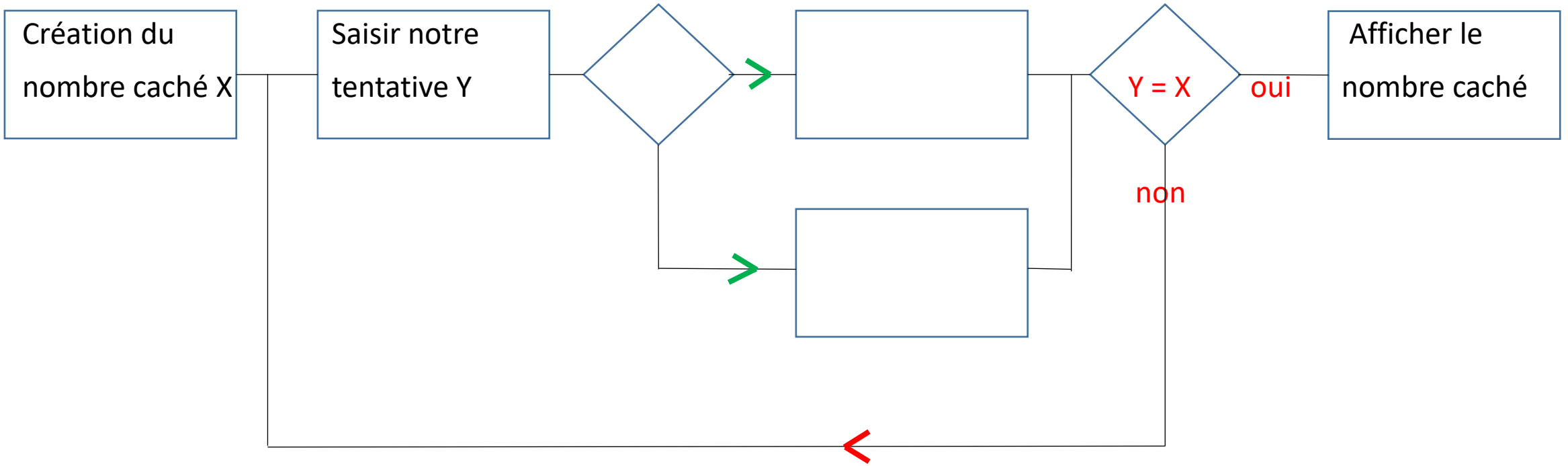
Où faire nos tentatives ?



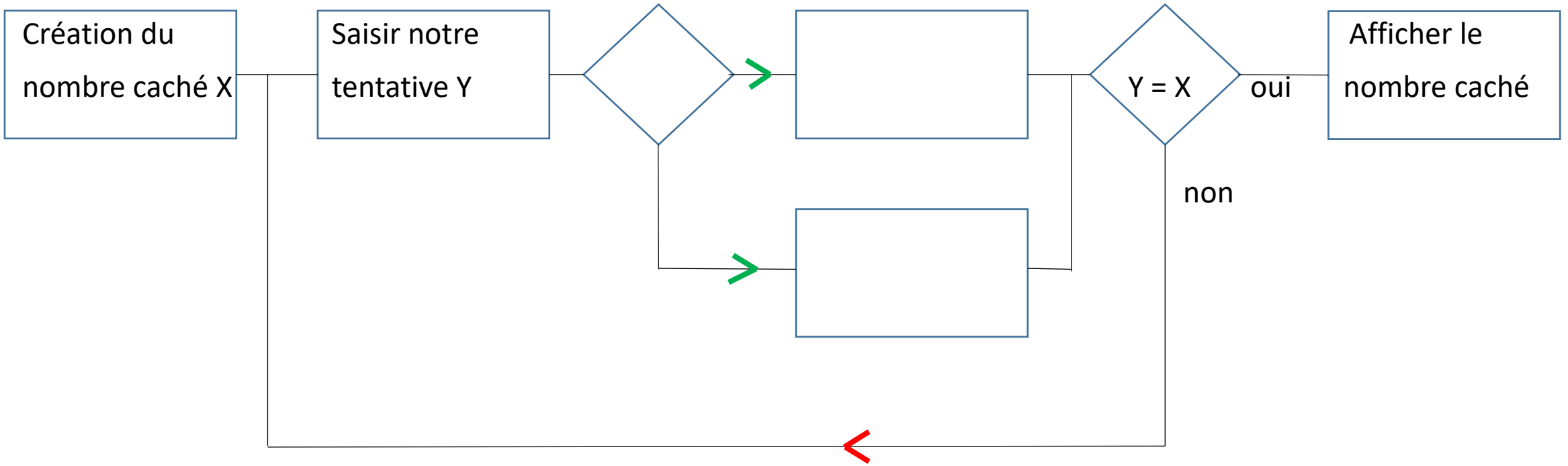
Quand sortir de la boucle ?



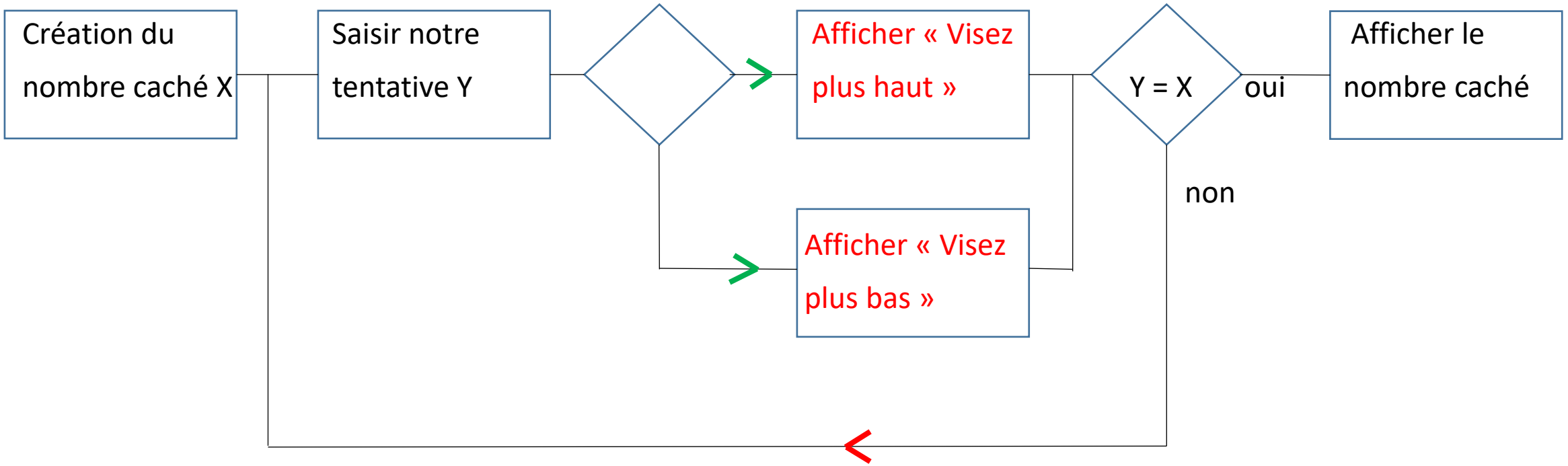
Quand sortir de la boucle ?



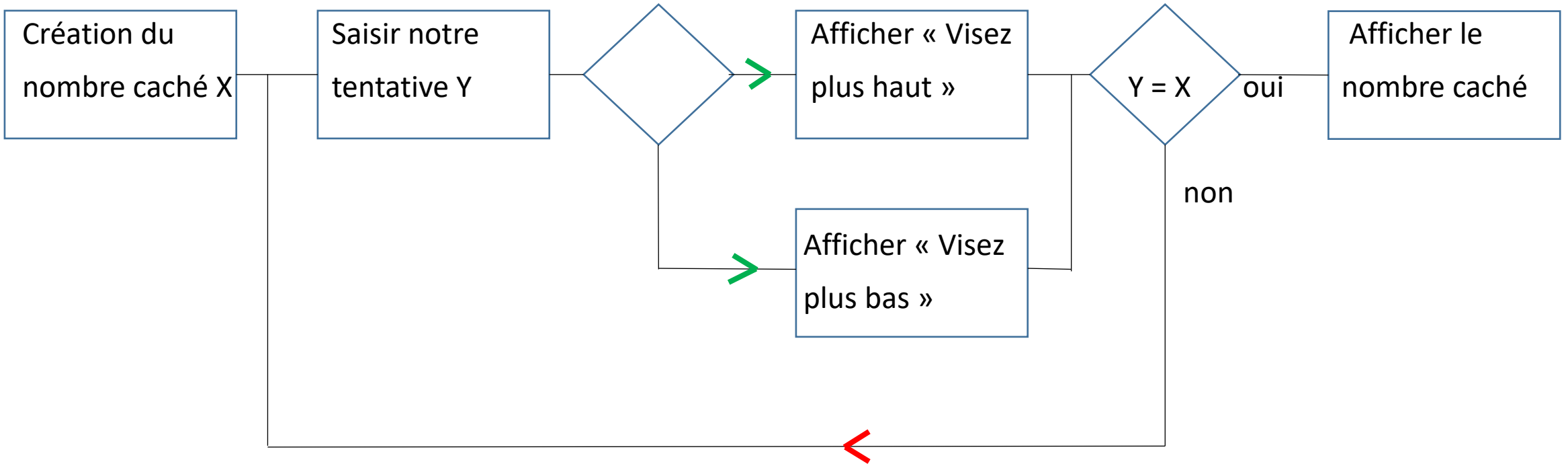
Quelles informations me sont données ?



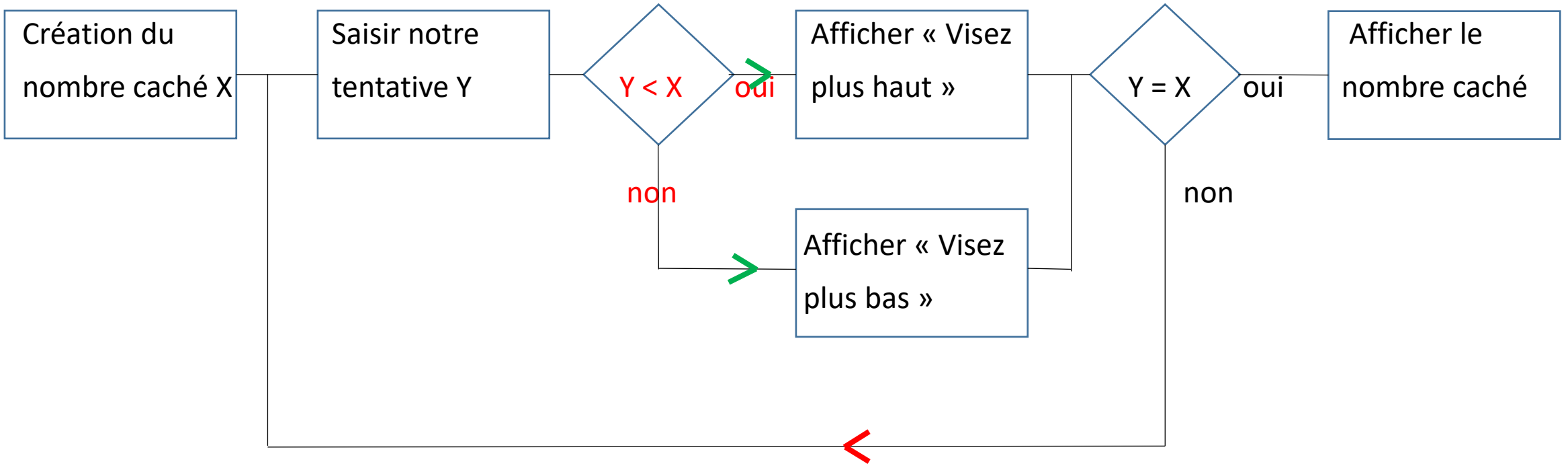
Quelles informations me sont données ?



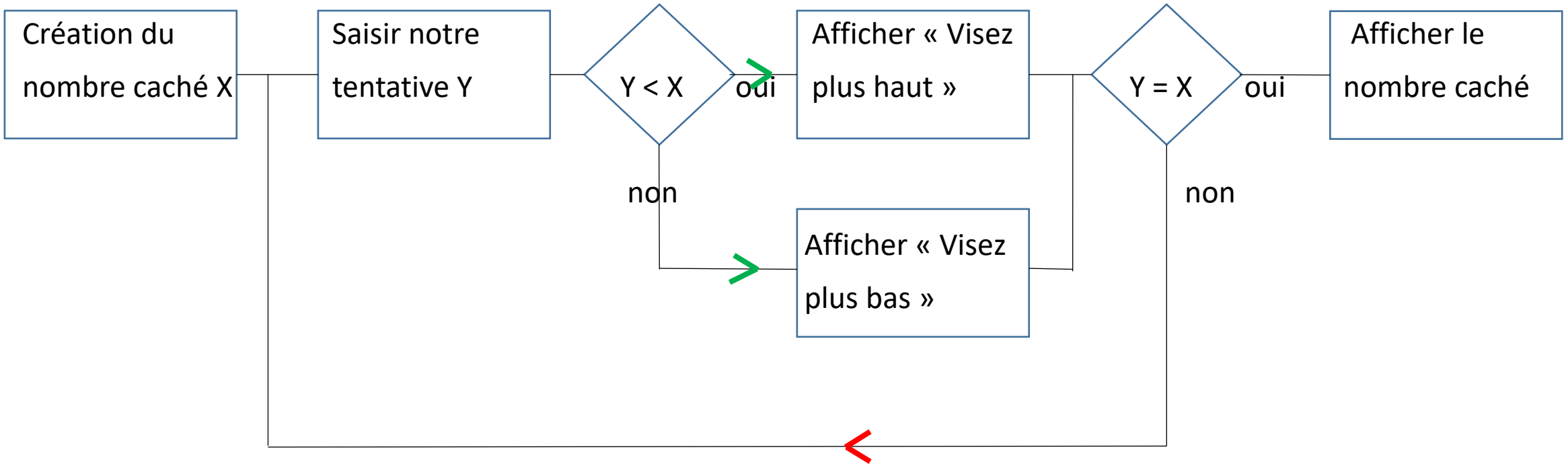
Selon quelle condition ?



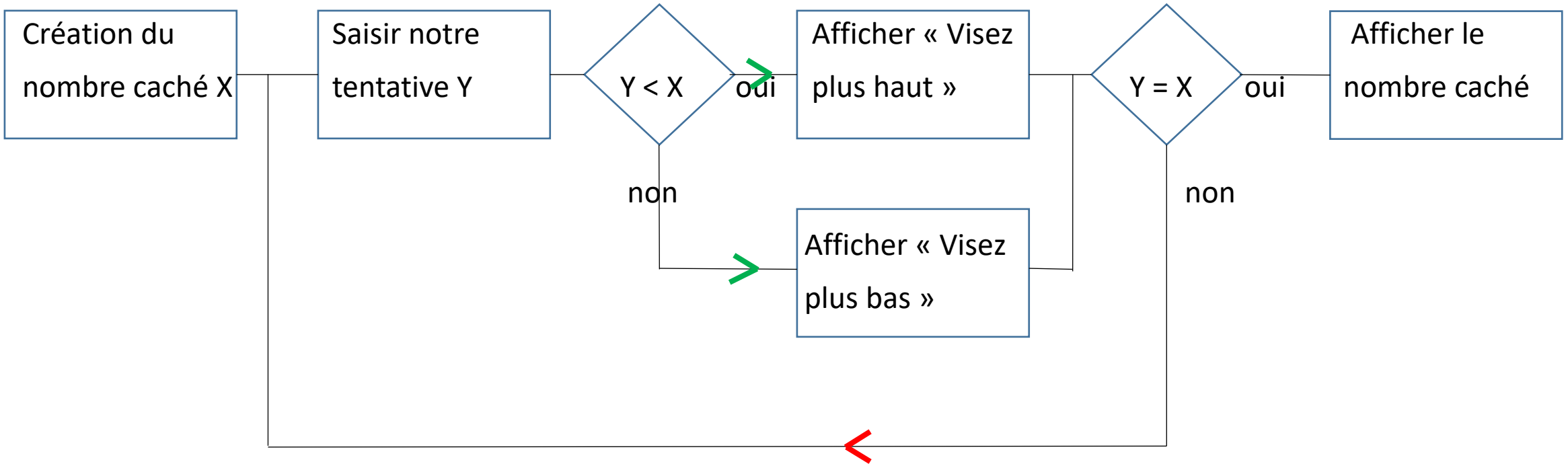
Selon quelle condition ?



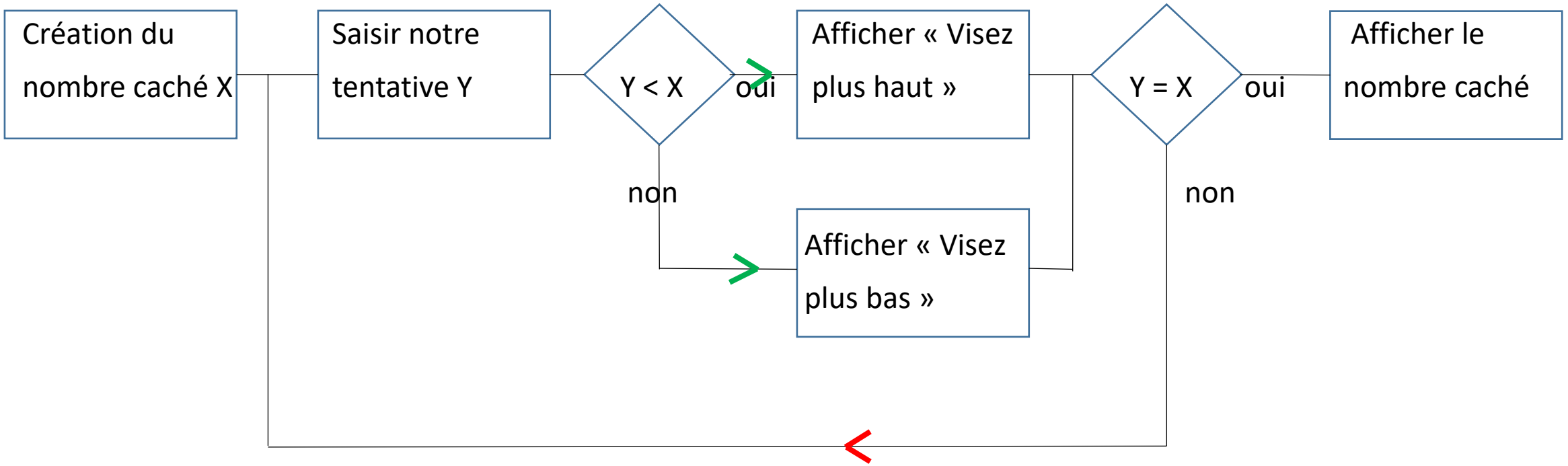
Quel est le défaut de cet organigramme ?



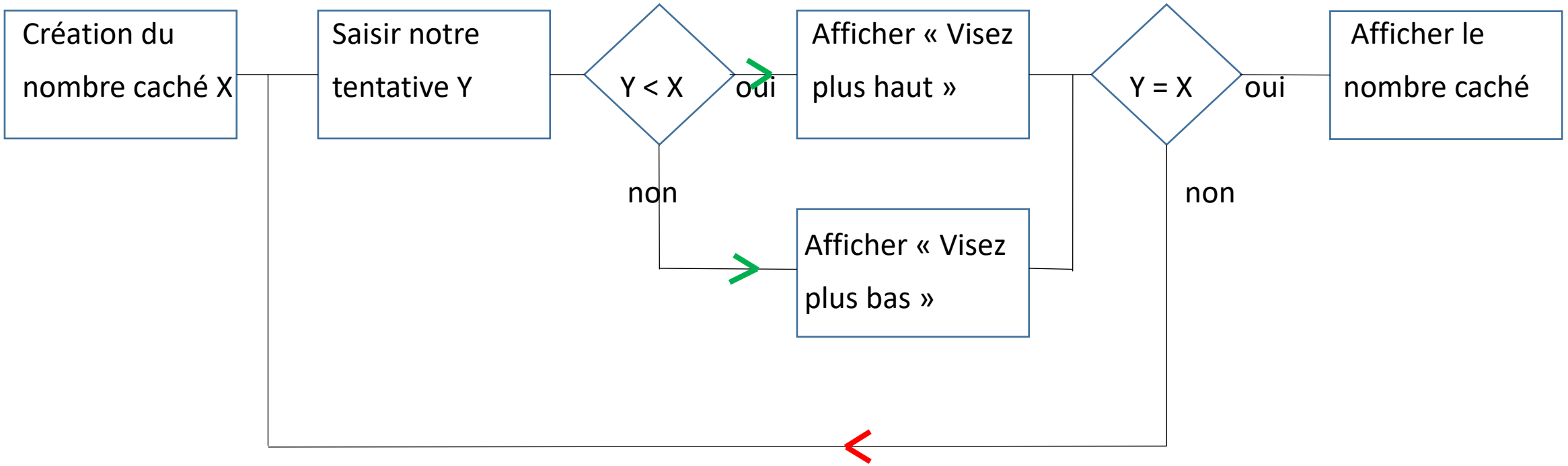
A la dernière tentative, la bonne…



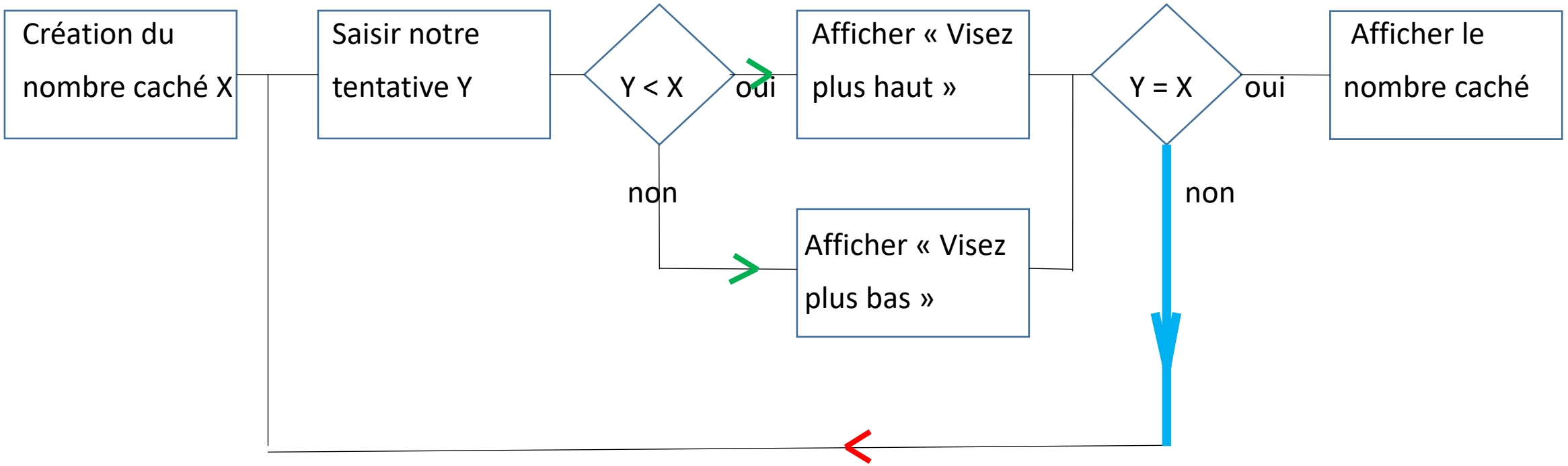
A la dernière tentative (la bonne), il va m’afficher « Viser plus bas » inutilement (puisque j’ai trouvé le nombre caché).



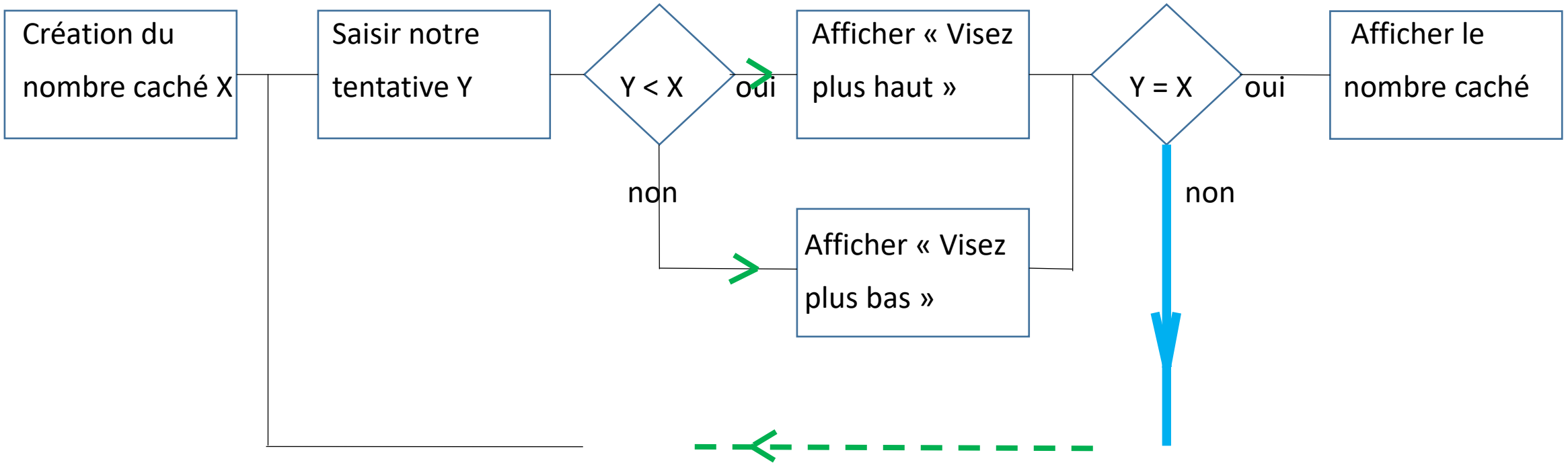
Modification : les informations doivent être placées <u>après</u> la condition de sortie de la boucle :



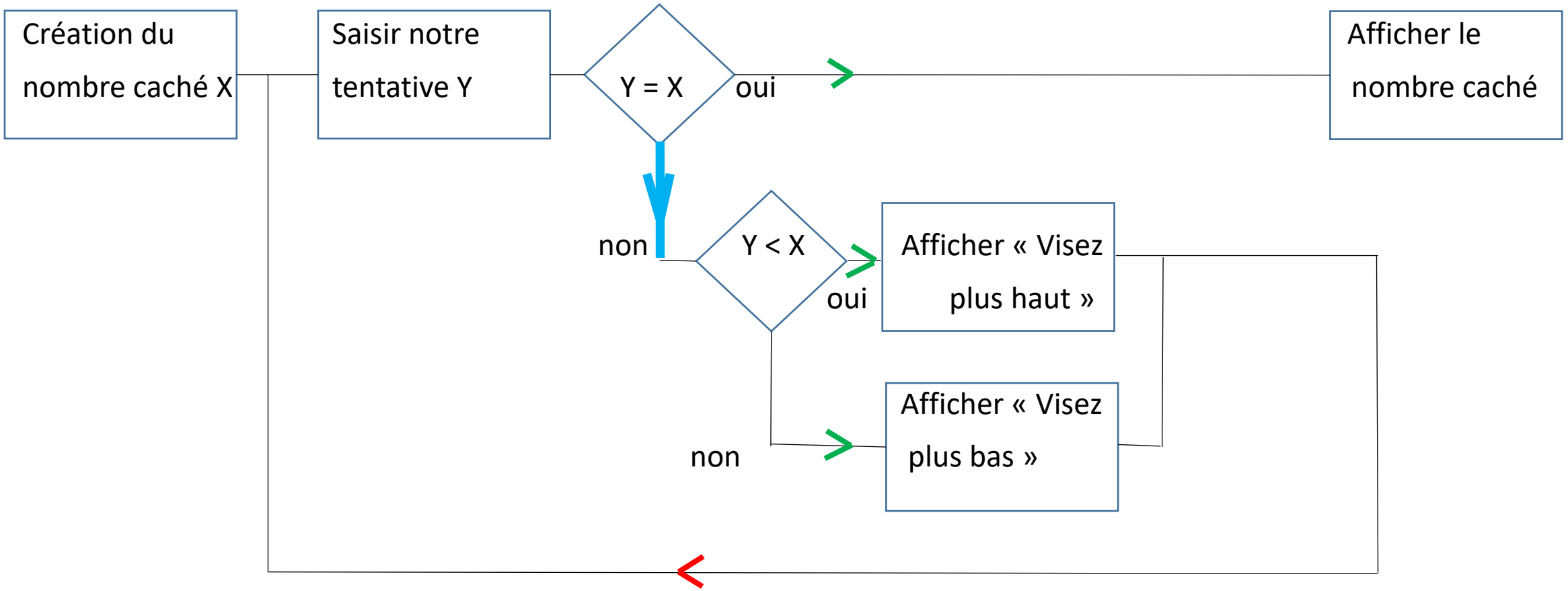
Modification : les informations doivent être placées <u>après</u> la condition de sortie de la boucle :



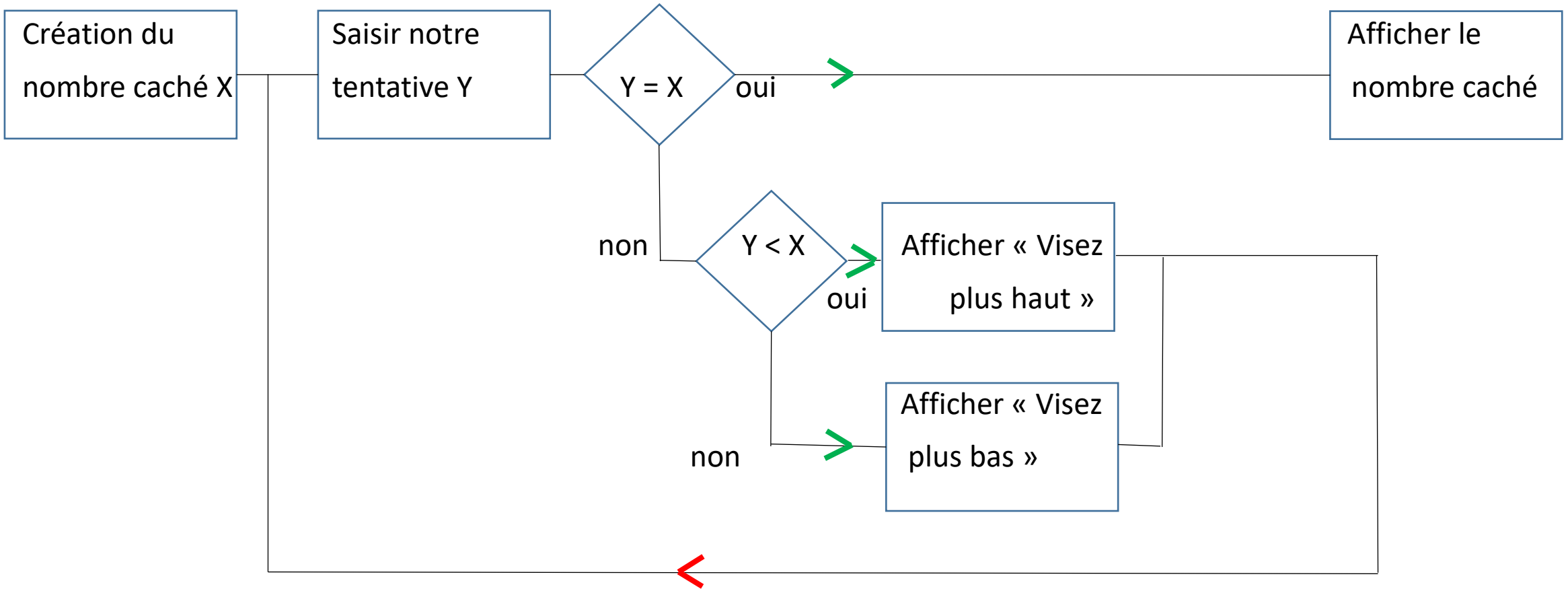
Modification : les informations doivent être placées <u>après</u> la condition de sortie de la boucle :

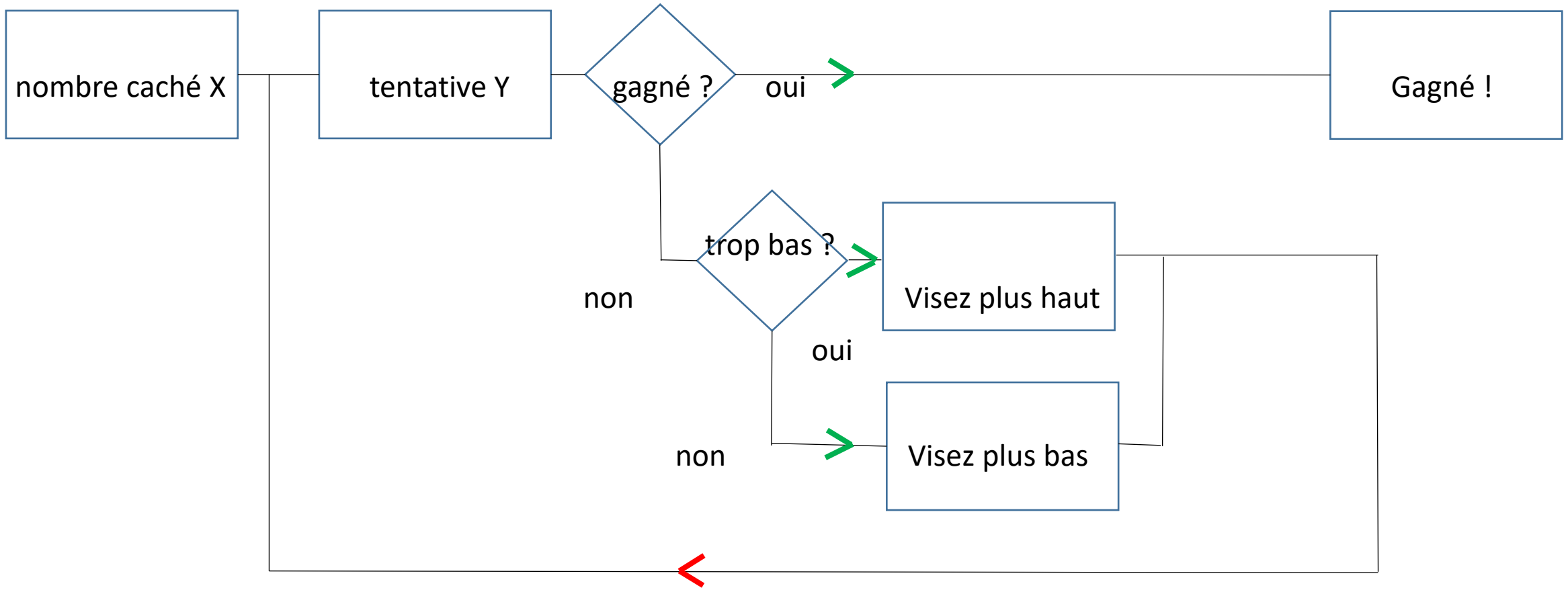


Modification : les informations doivent être placées <u>après</u> la condition de sortie de la boucle :

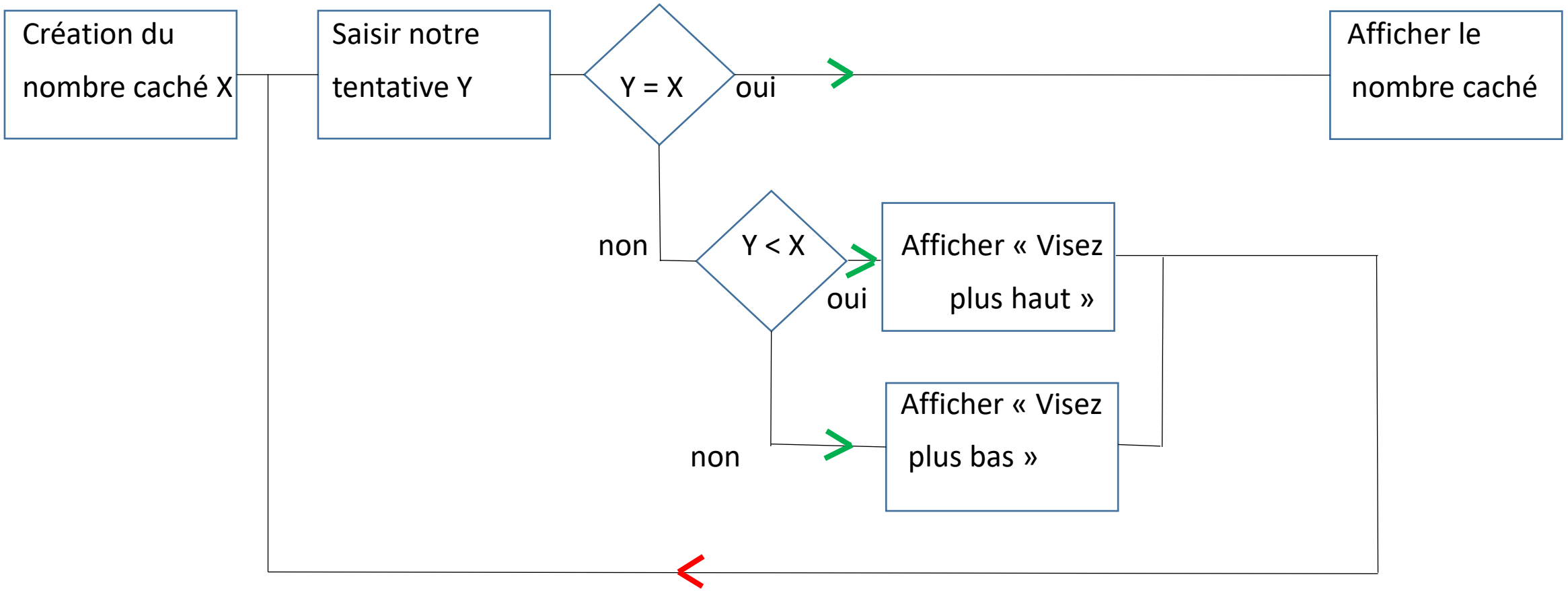


Cet organigramme n’a pas le (petit) défaut (qui n’arrive qu’une fois) de m’afficher « Visez… » lorsque j’ai trouvé le nombre caché :

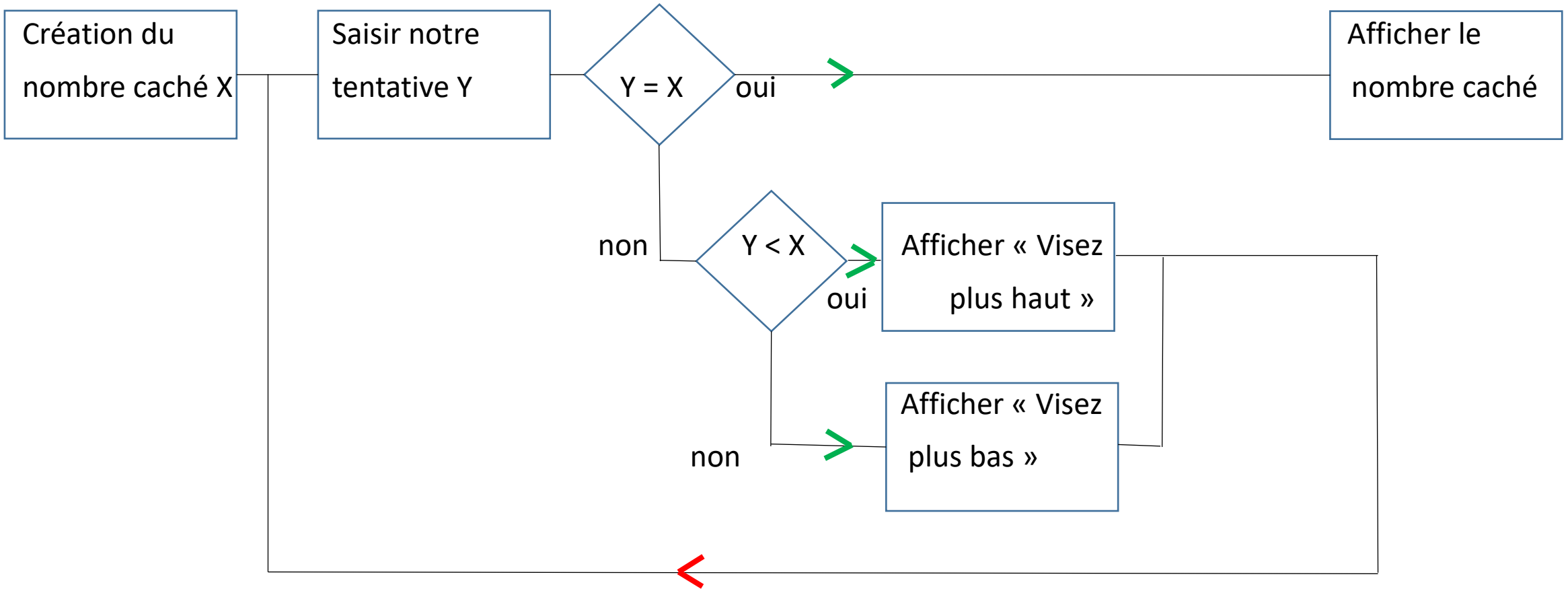



nombre caché X
tentative Y
gagné ?
oui
Gagné !
trop bas ?
non
Visez plus haut
oui
non
Visez plus bas

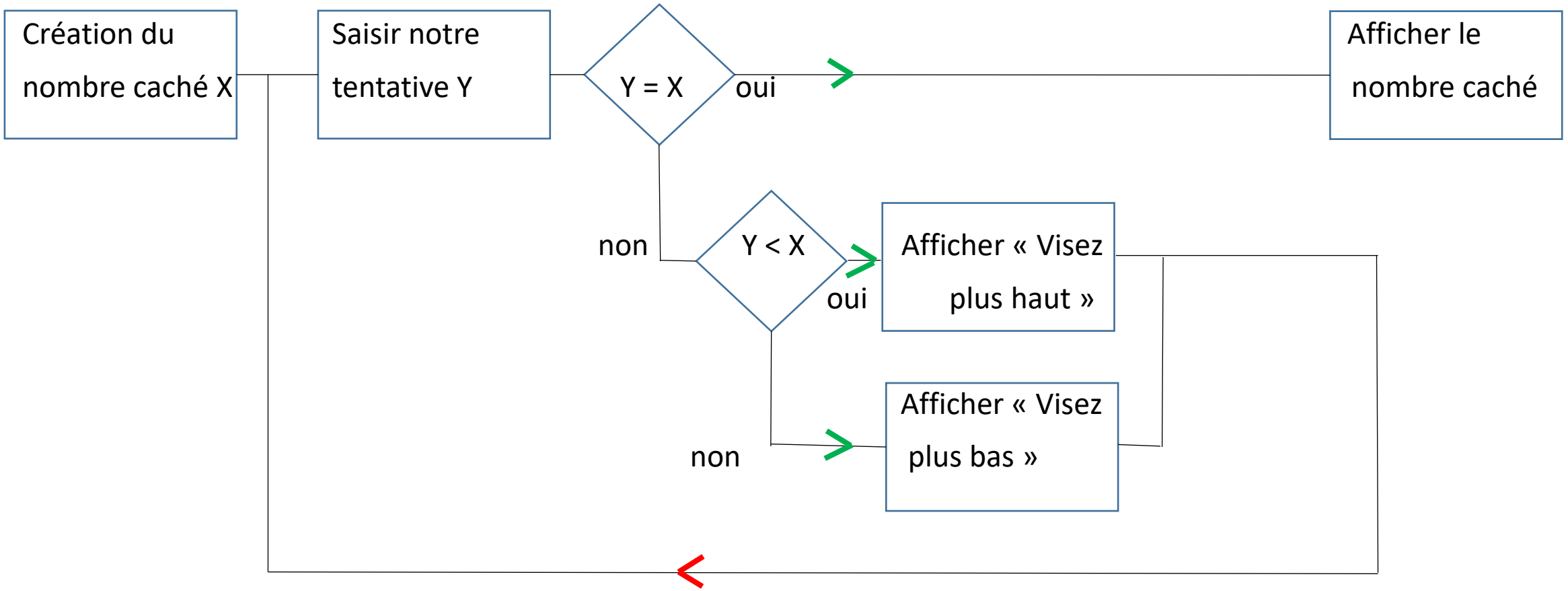
Quelle demande de l’énoncé n’a pas eu sa réponse ?



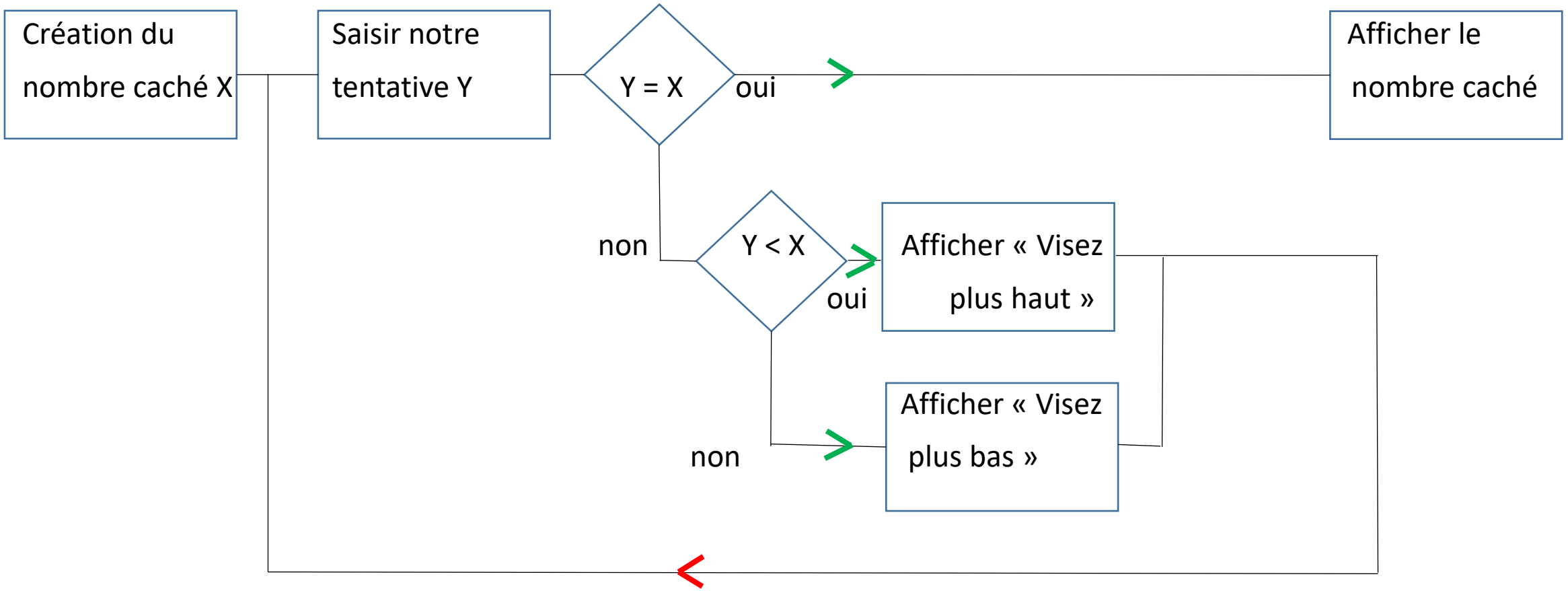
Connaître le nombre de tentatives !



Le nombre de tentatives correspondra dans l'organigramme au …

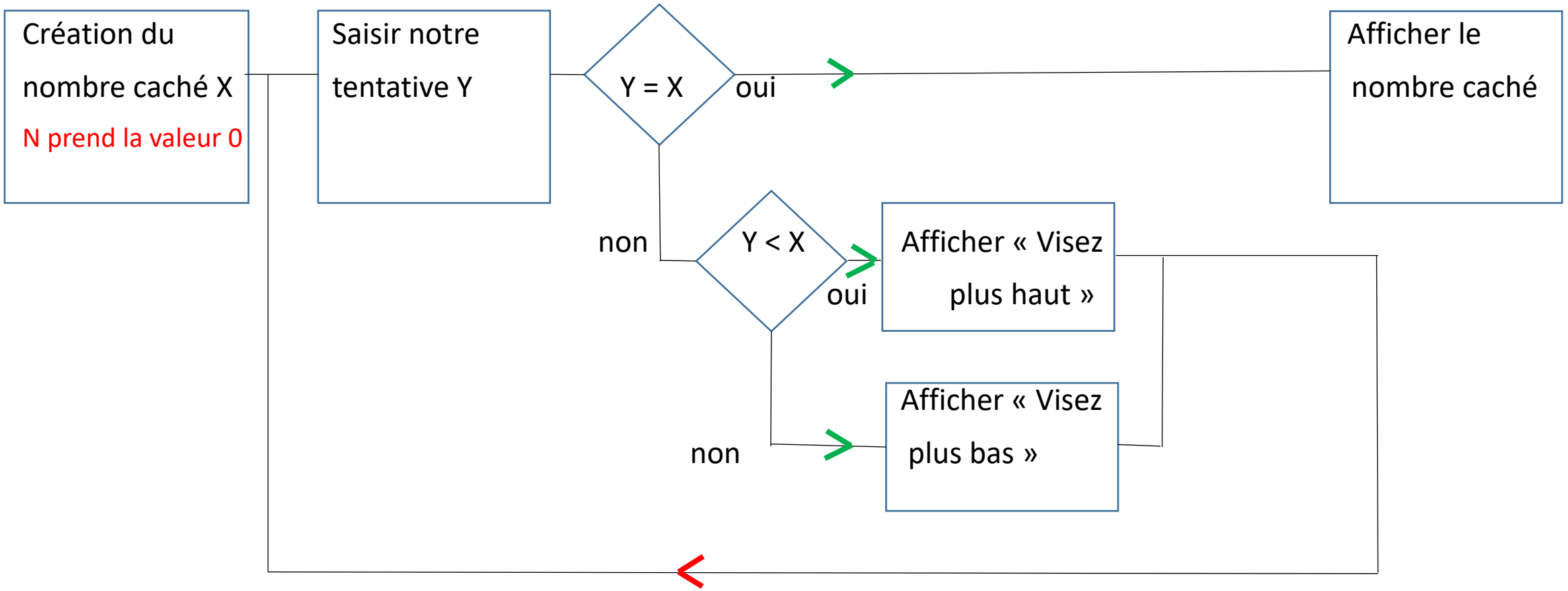


Le nombre de tentatives correspondra dans l'organigramme au nombre de fois où l'on est passé dans la boucle.



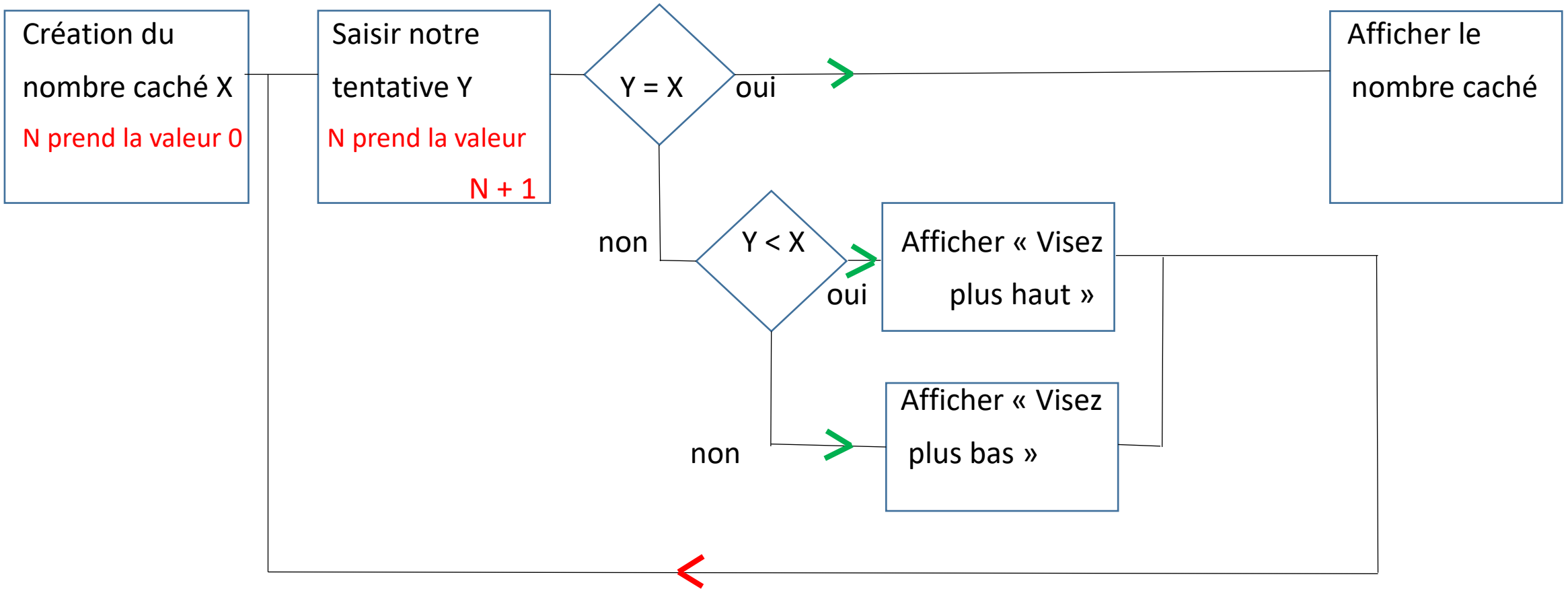
Le nombre de tentatives correspondra dans l'organigramme au nombre de fois où l'on est passé dans la boucle.

On va créer un compteur de boucle.



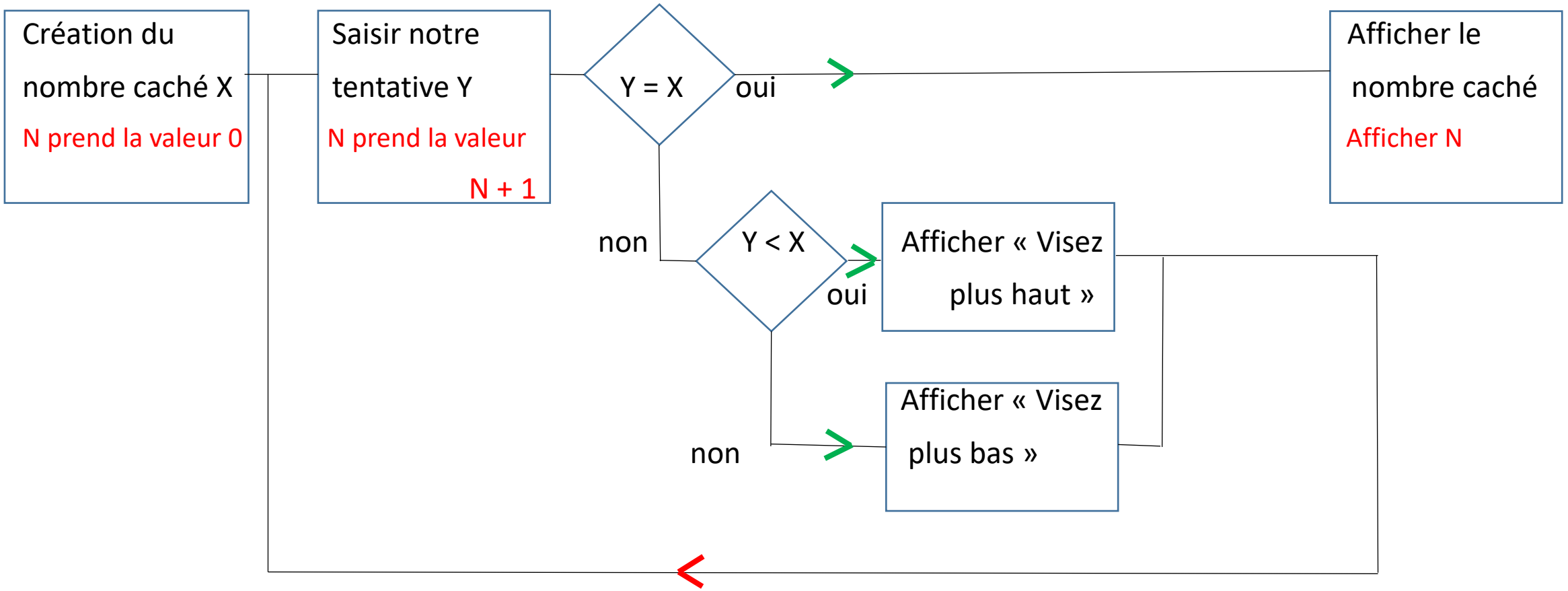
Le nombre de tentatives correspondra dans l'organigramme au nombre de fois où l'on est passé dans la boucle.

On va créer un compteur de boucle.



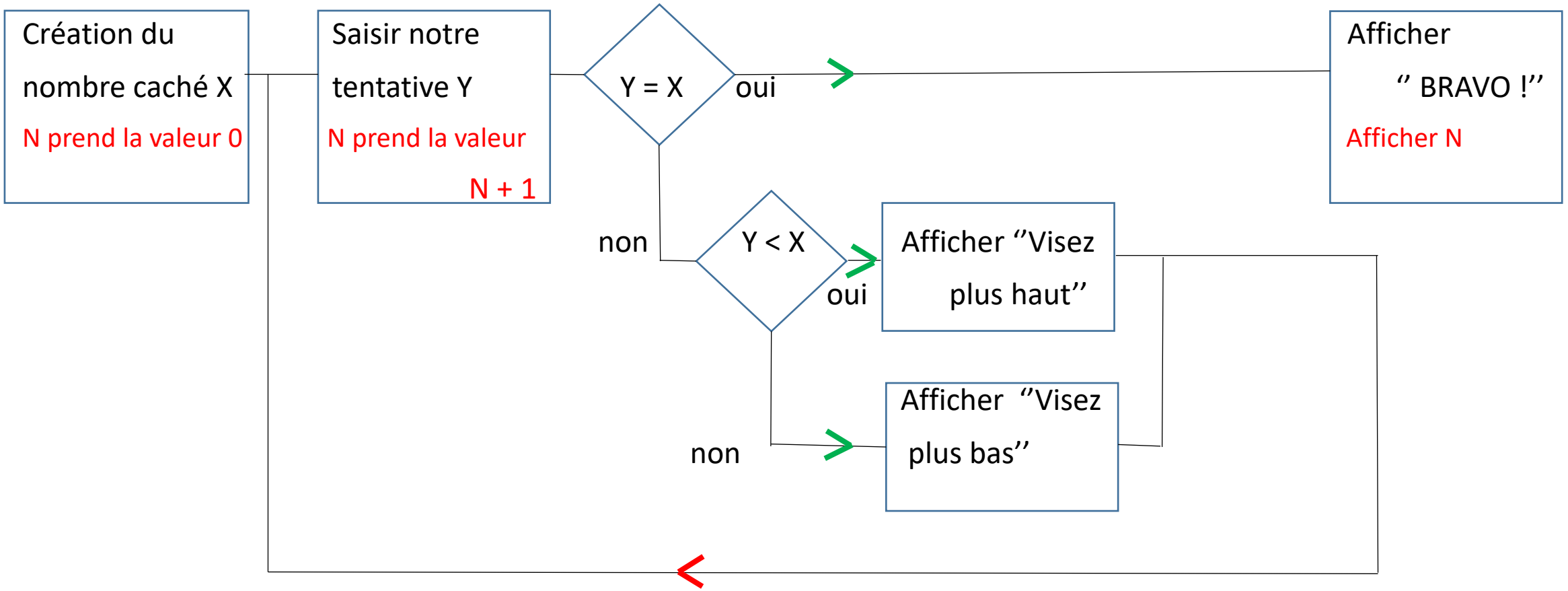
Le nombre de tentatives correspondra dans l'organigramme au nombre de fois où l'on est passé dans la boucle.

On va créer un compteur de boucle.

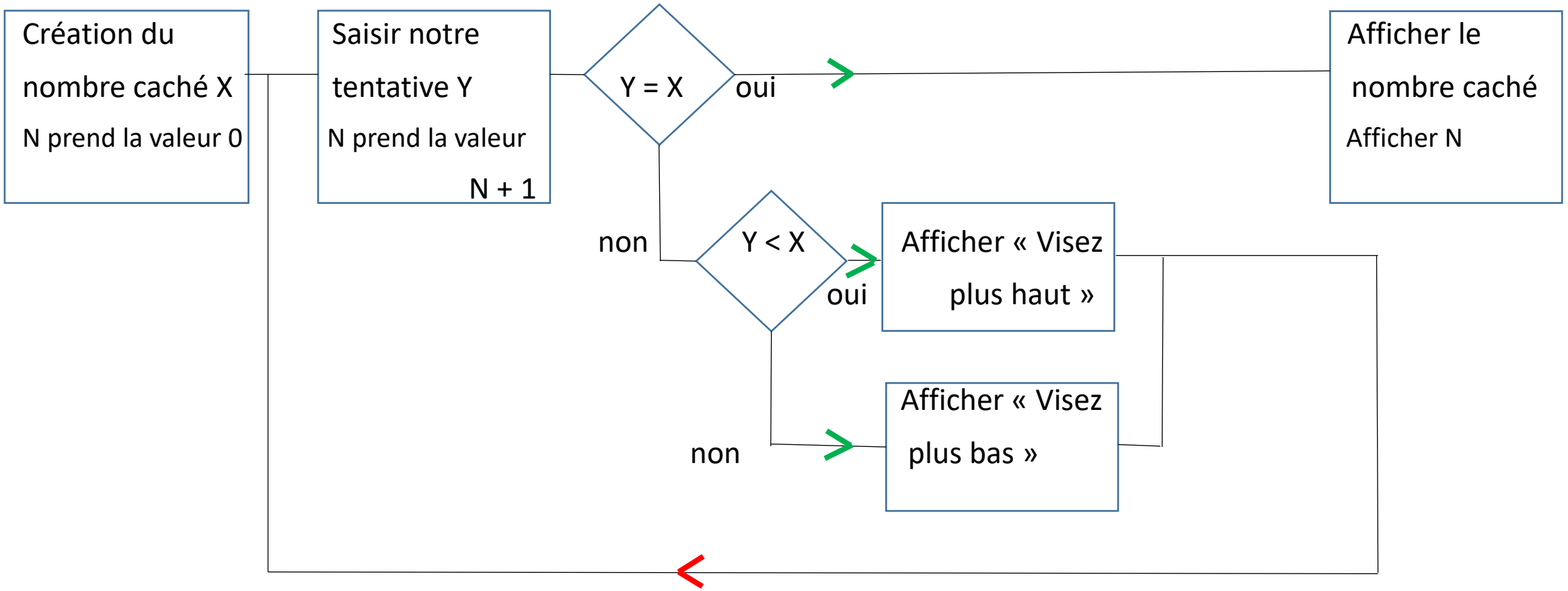


Le nombre de tentatives correspondra dans l'organigramme au nombre de fois où l'on est passé dans la boucle.

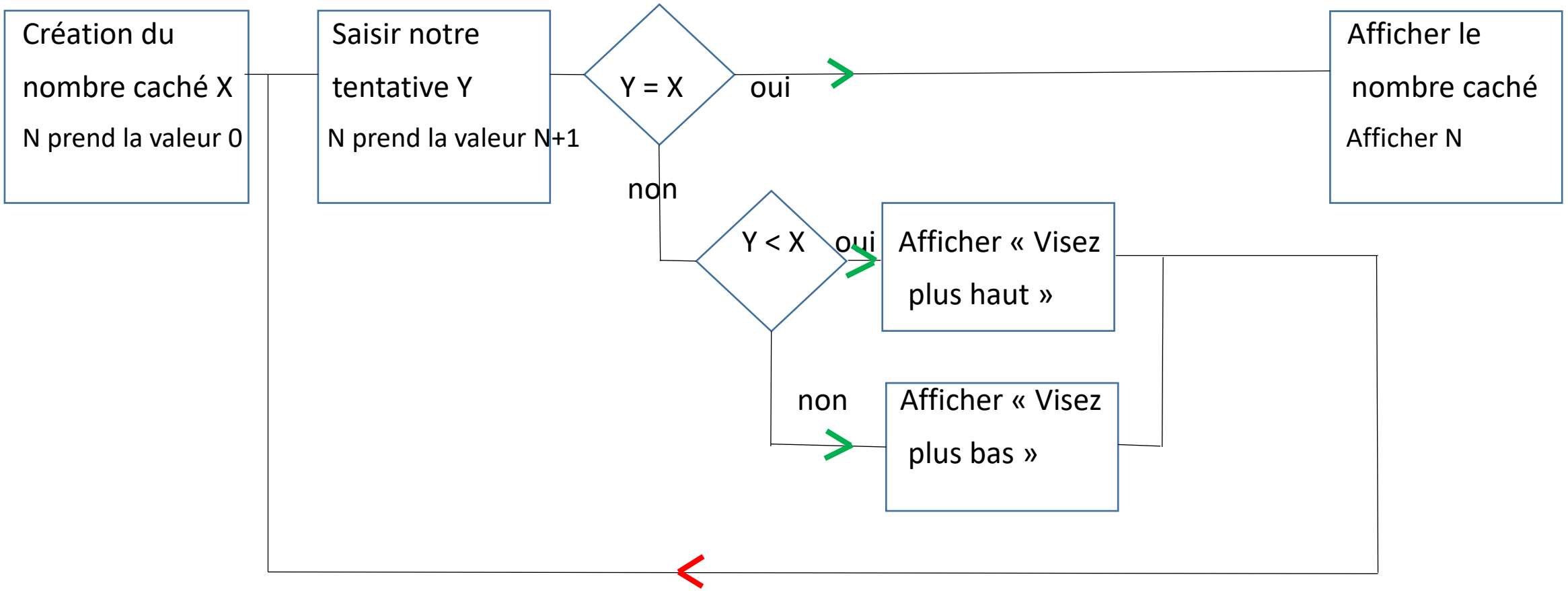
On va créer un compteur de boucle.



Etape 2 : …


Création du nombre caché X
N prend la valeur 0
Saisir notre tentative Y
N prend la valeur N + 1
Y = X
oui
Afficher le nombre caché
Afficher N
non
Y < X
oui
Afficher « Visez plus haut »
non
Afficher « Visez plus bas »

Etape 2 : écriture du programme sur une feuille.



Fonctionnalités utiles de la machine :

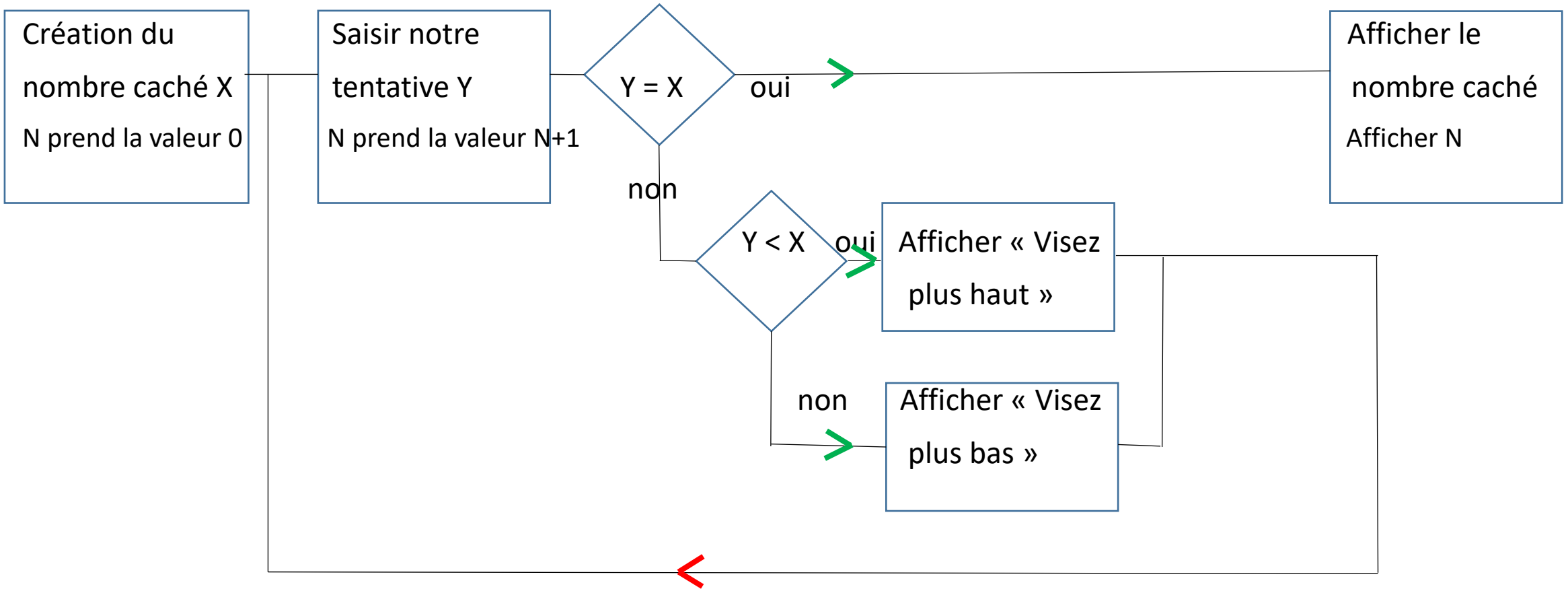
on utilisera que les fonctionnalités « If Then Else Goto », car :

elles permettent de faire fonctionner tous les types d'algorithme ;

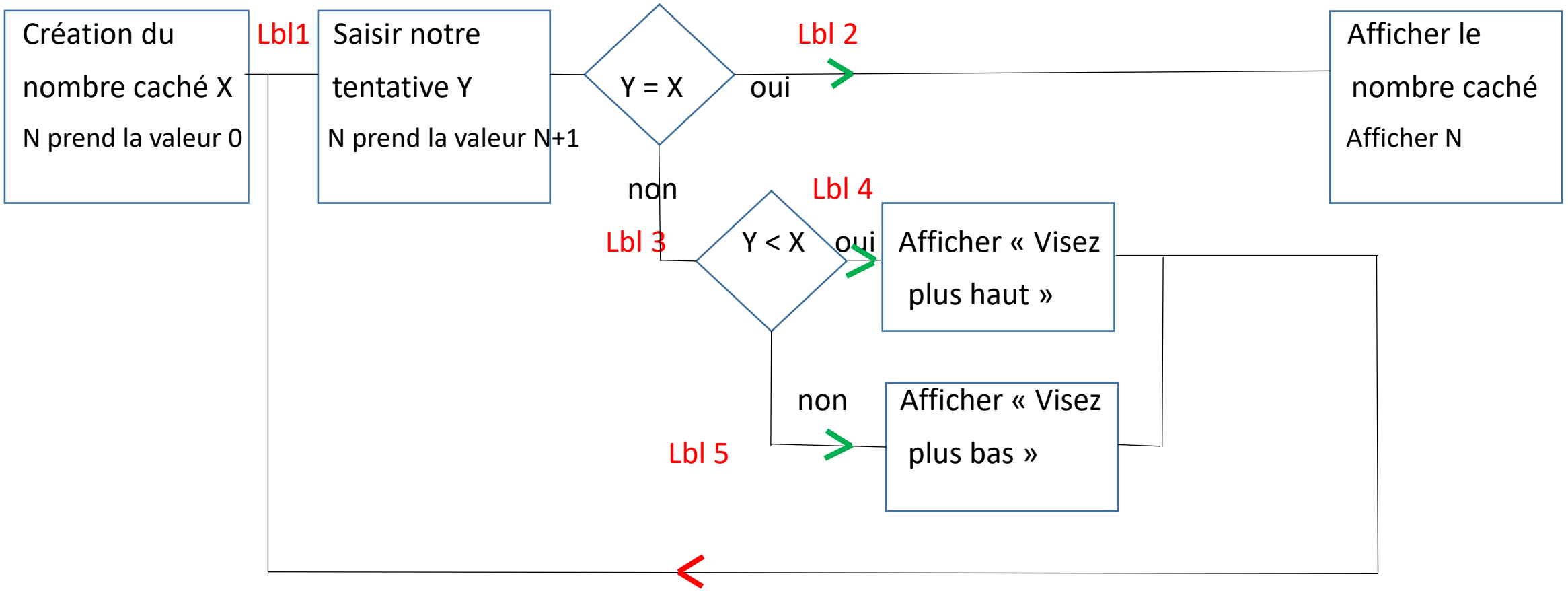
elles permettent de traduire tous les types d'organigramme ;

elles seules permettent à **un élève** de démontrer qu'il possède les connaissances (créer une boucle) et pas seulement d'utiliser les connaissances de **la calculatrice** (qui possède des fonctionnalités correspondant à une boucle comme « Tant que », « Jusqu'à » …).

Etape 2 : écriture du programme sur une feuille. On ajoute …



Etape 2 : écriture du programme sur une feuille. On ajoute des adresses sur l'organigramme.



Création d’un nombre entier aléatoire :

RAN# (dans OPTN → PROBA → RAND) crée un nombre aléatoire dans [0 ; 1].

Création d’un nombre entier aléatoire :

RAN# (dans OPTN → PROBA → RAND) crée un nombre aléatoire dans [0 ; 1].

Comment en créer un entre 0 et 1000 ?

Création d'un nombre entier aléatoire :

RAN# (dans OPTN → PROBA → RAND) crée un nombre aléatoire dans [0 ; 1].

Comment en créer un entre 0 et 1000 ?

1000 × RAN# crée un nombre aléatoire dans [0 ; 1000].

Création d'un nombre entier aléatoire :

RAN# (dans OPTN → PROBA → RAND) crée un nombre aléatoire dans [0 ; 1].

Comment en créer un entre 0 et 1000 ?

1000 × RAN# crée un nombre aléatoire dans [0 ; 1000].

Comment créer un nombre entier dans [0 ; 1000] ?

Création d'un nombre entier aléatoire :

RAN# (dans OPTN → PROBA → RAND) crée un nombre aléatoire dans [0 ; 1].

Comment en créer un entre 0 et 1000 ?

1000 × RAN# crée un nombre aléatoire dans [0 ; 1000].

Comment créer un nombre entier dans [0 ; 1000] ?

Int (1000 × RAN#) crée un nombre aléatoire dans { 0 ; 1 ; … ; 1000 }.

Ent ou Int signifie « *partie entière* » et se trouve dans OPTN → NUM.

Création d’un nombre entier aléatoire :

Exemple :

comment créer un nombre aléatoire entier entre 500 et 700 ?

Création d’un nombre entier aléatoire :

Exemple :

comment créer un nombre aléatoire entier entre 500 et 700 ?

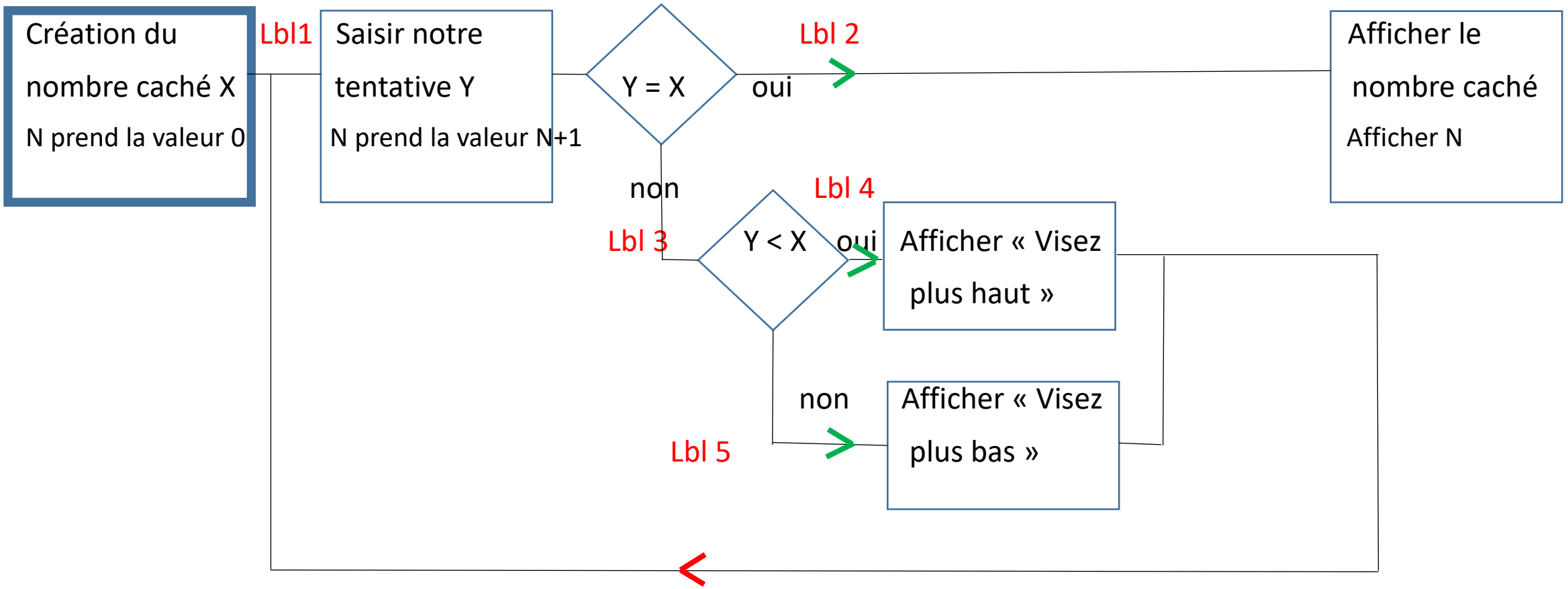
RAN# est entre 0 et 1

200 × RAN# est entre 0 et 200

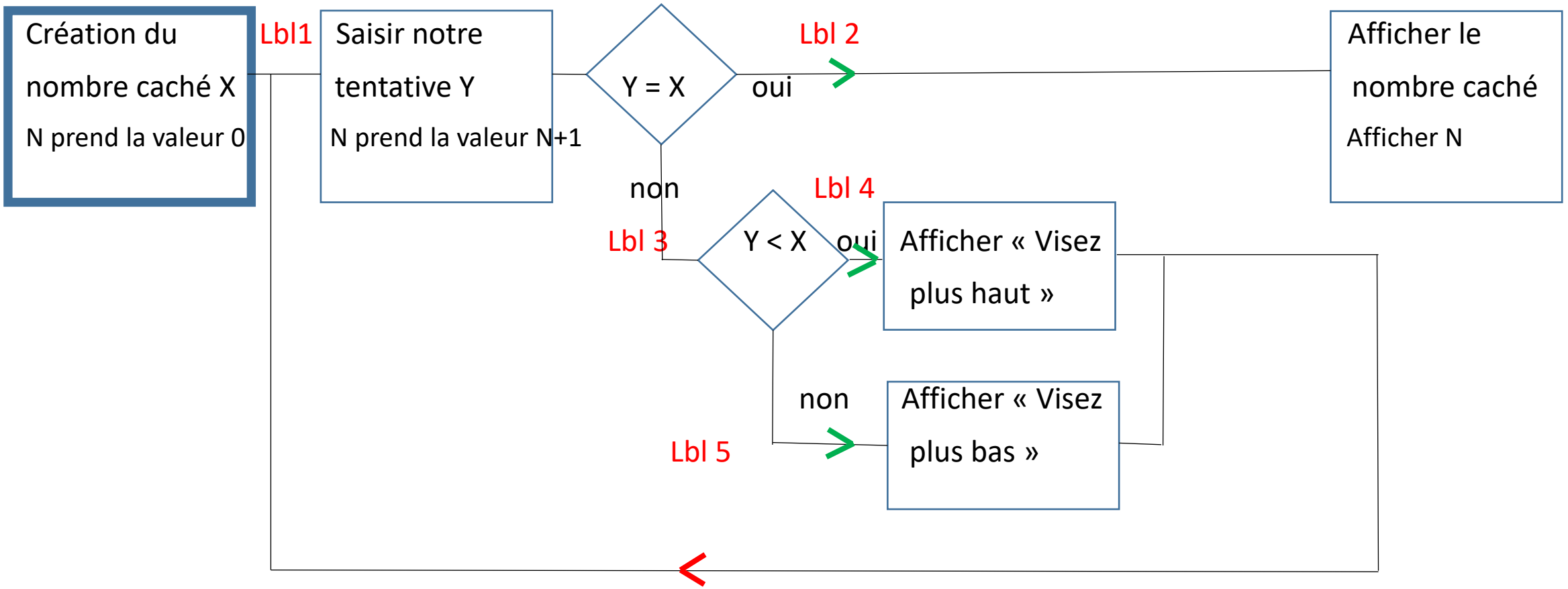
500 + (200 × RAN#)

est entre 500 et 700

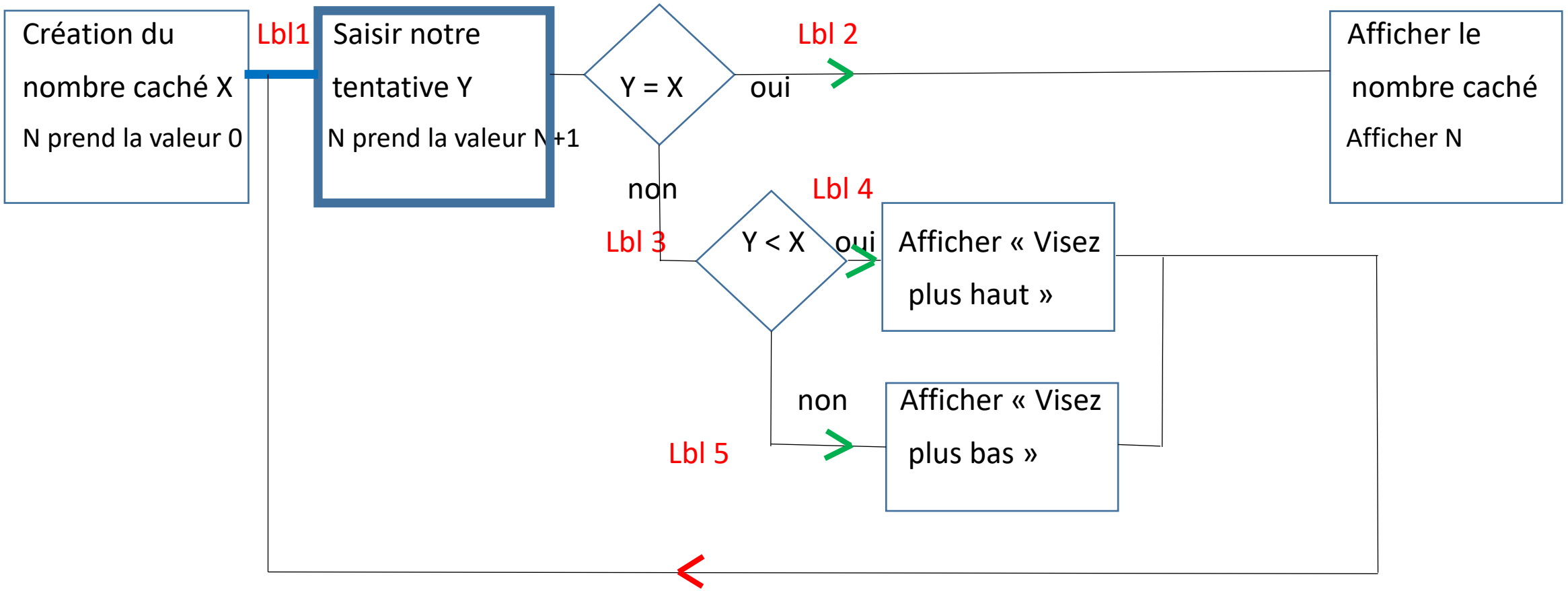
...


Création du nombre caché X
N prend la valeur 0
Lbl1
Saisir notre tentative Y
N prend la valeur N+1
Y = X
oui
Lbl 2
Afficher le nombre caché
Afficher N
non
Lbl 4
Lbl 3
Y < X
oui
Afficher « Visez plus haut »
non
Afficher « Visez plus bas »
Lbl 5

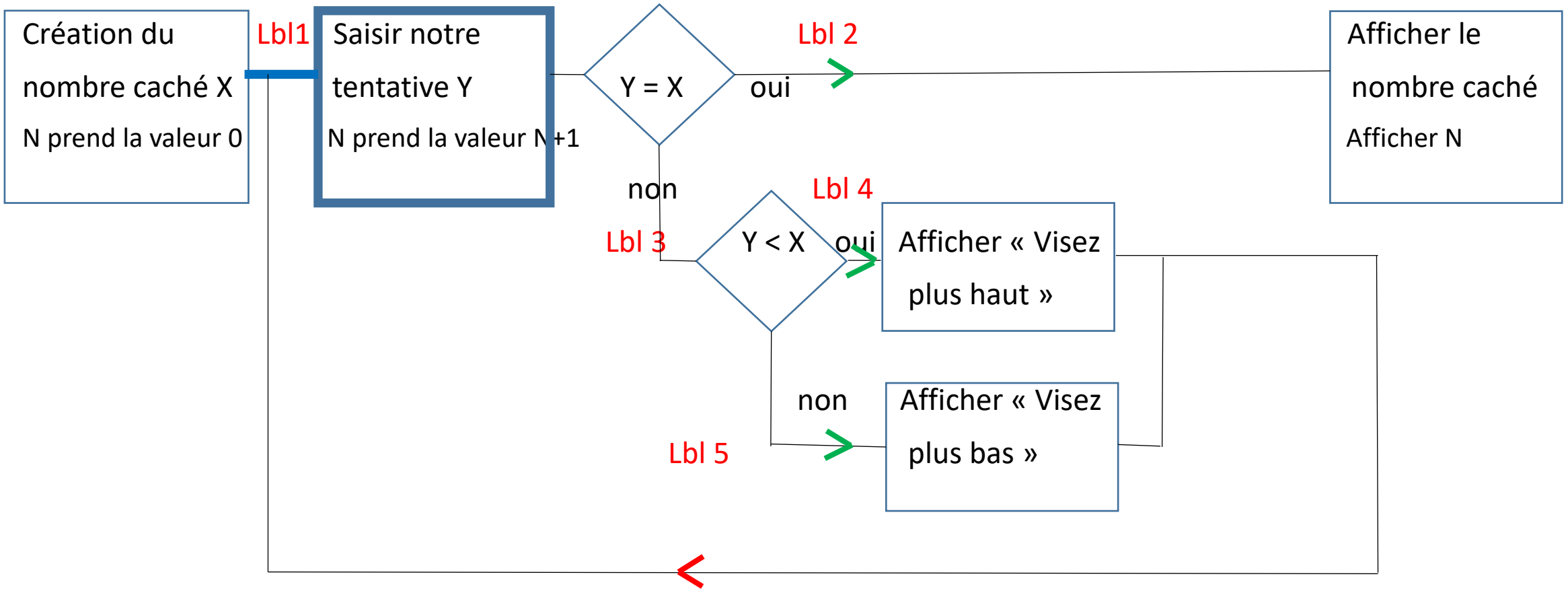
Int(1000×RAN#)→X : 0→N



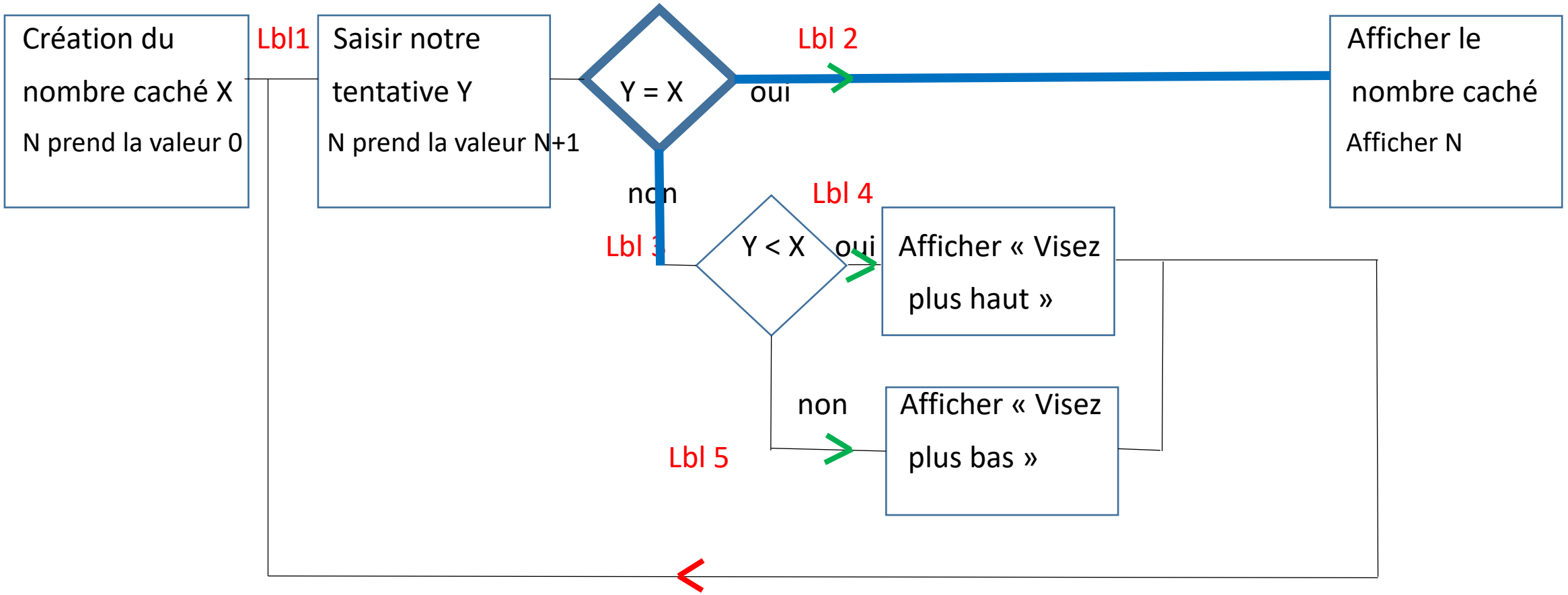
Int(1000×RAN#)→X : 0→N :



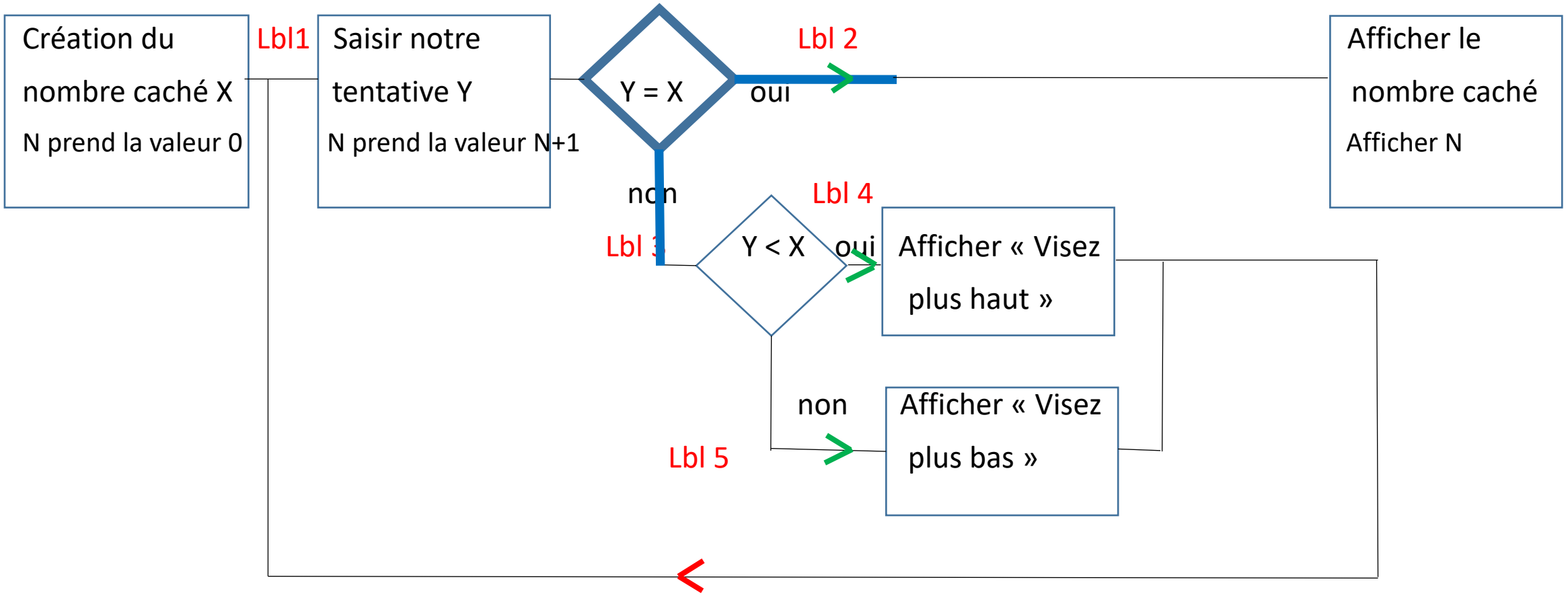
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N



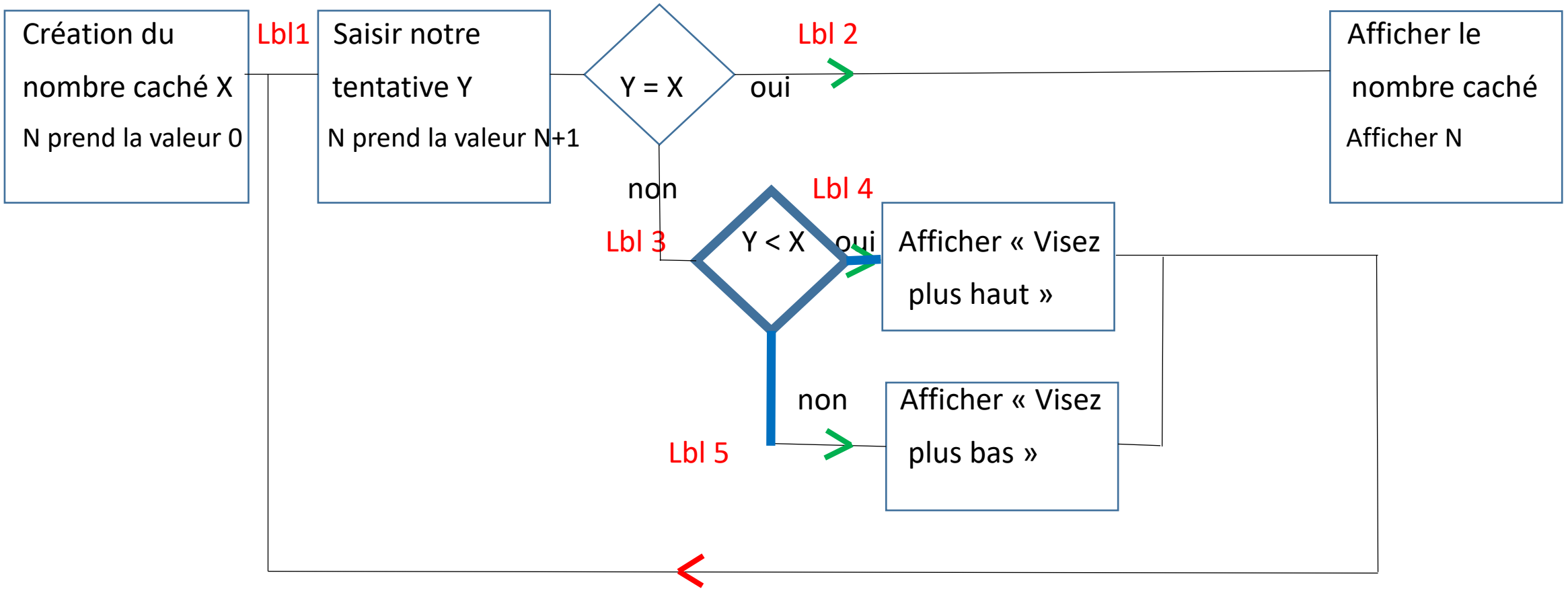
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N :



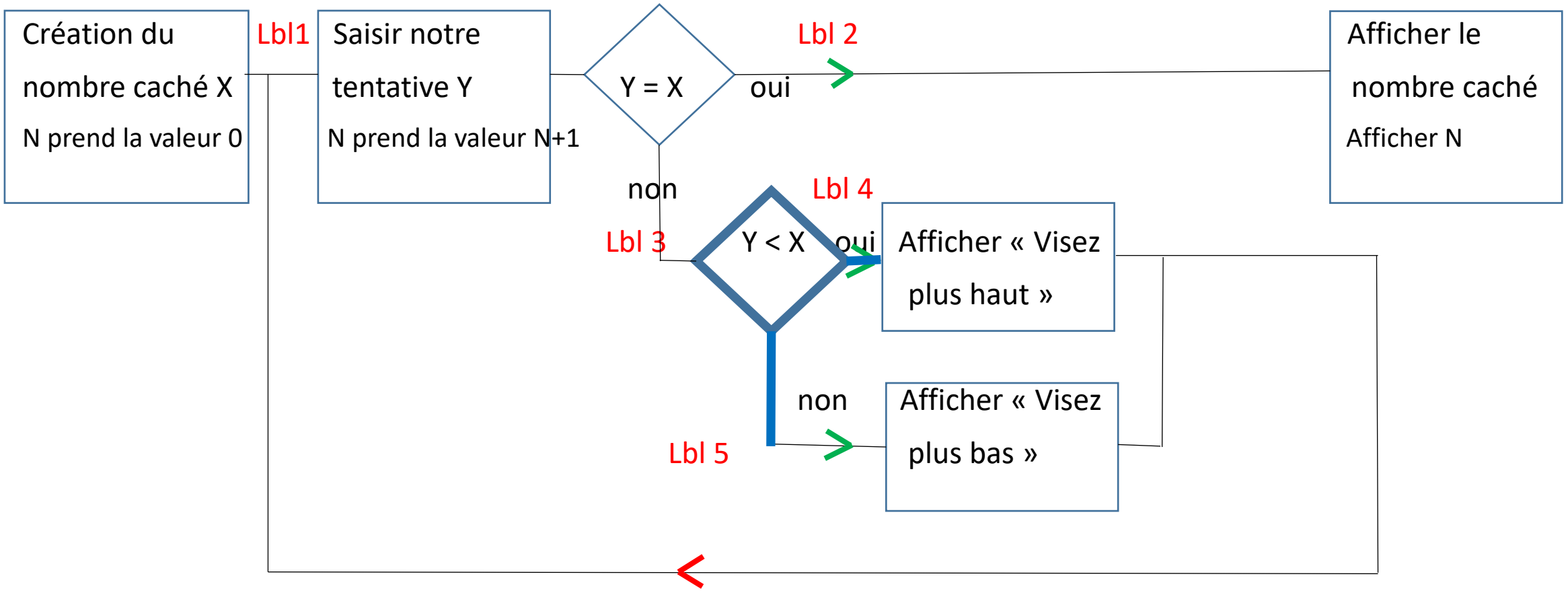
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3:



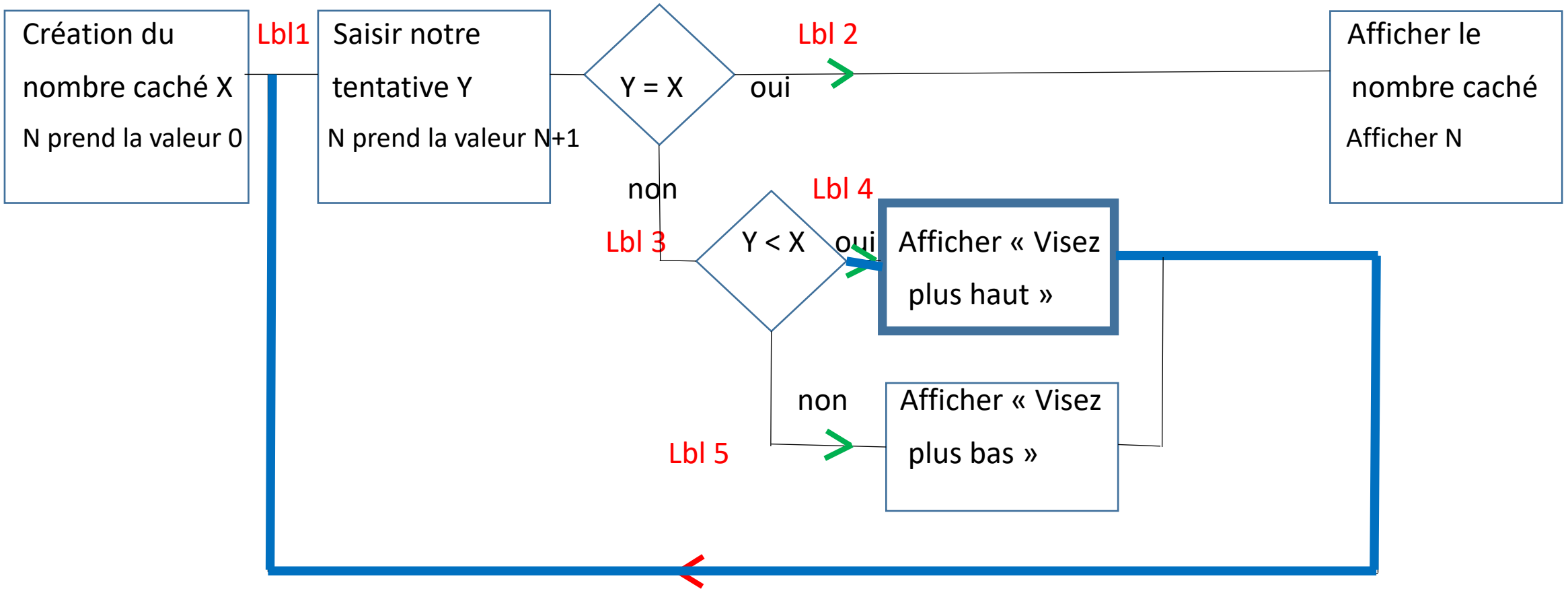
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3:



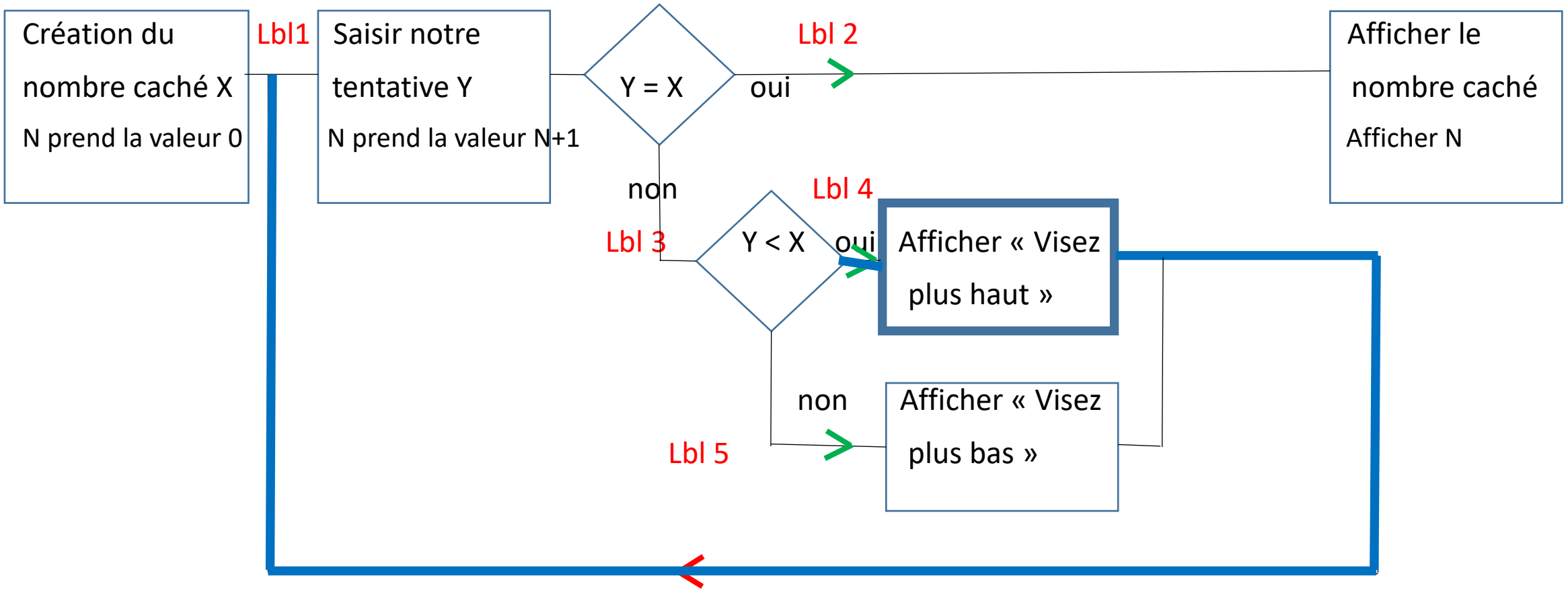
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 :



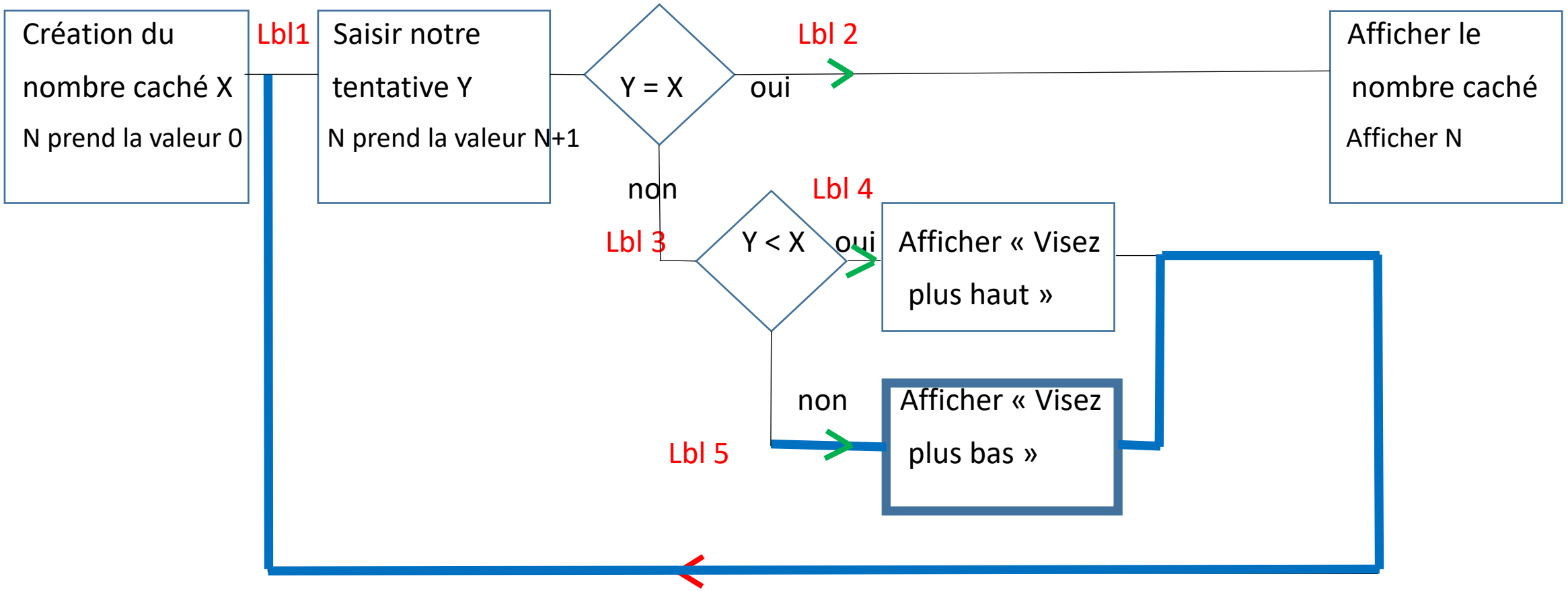
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 :



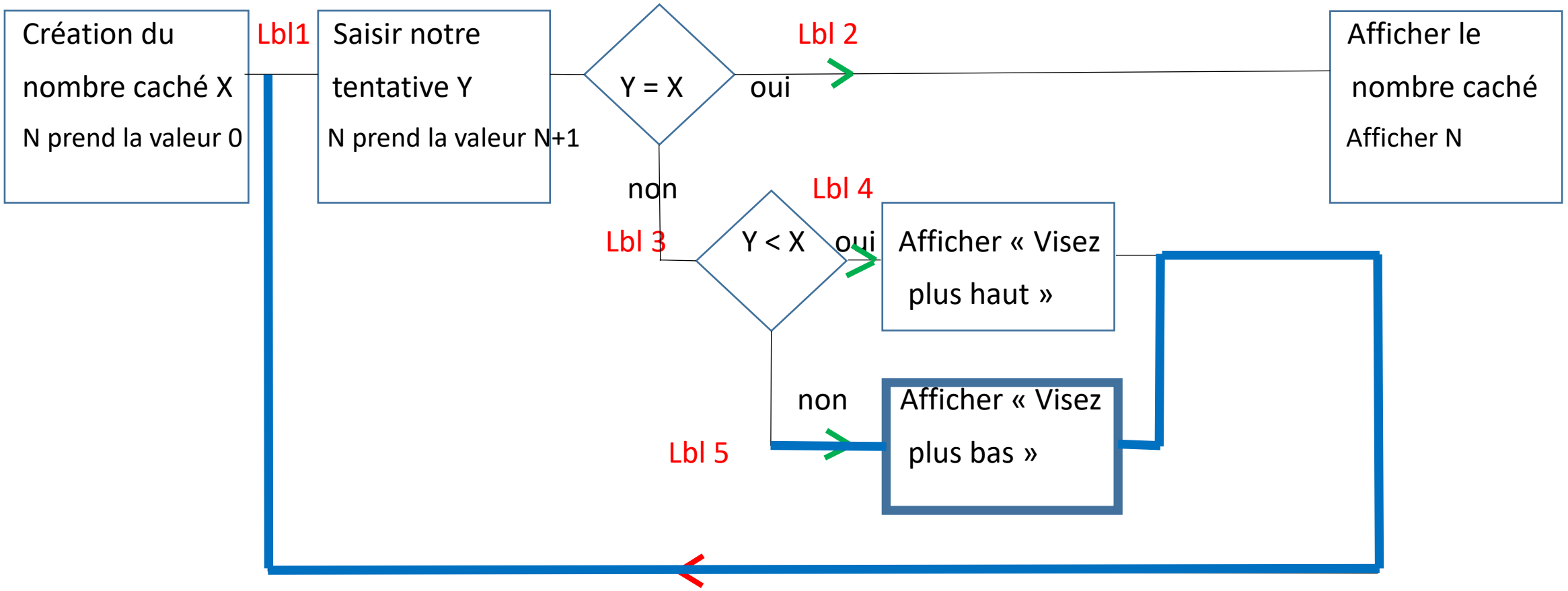
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 : Lbl4 : "+": Goto1 :



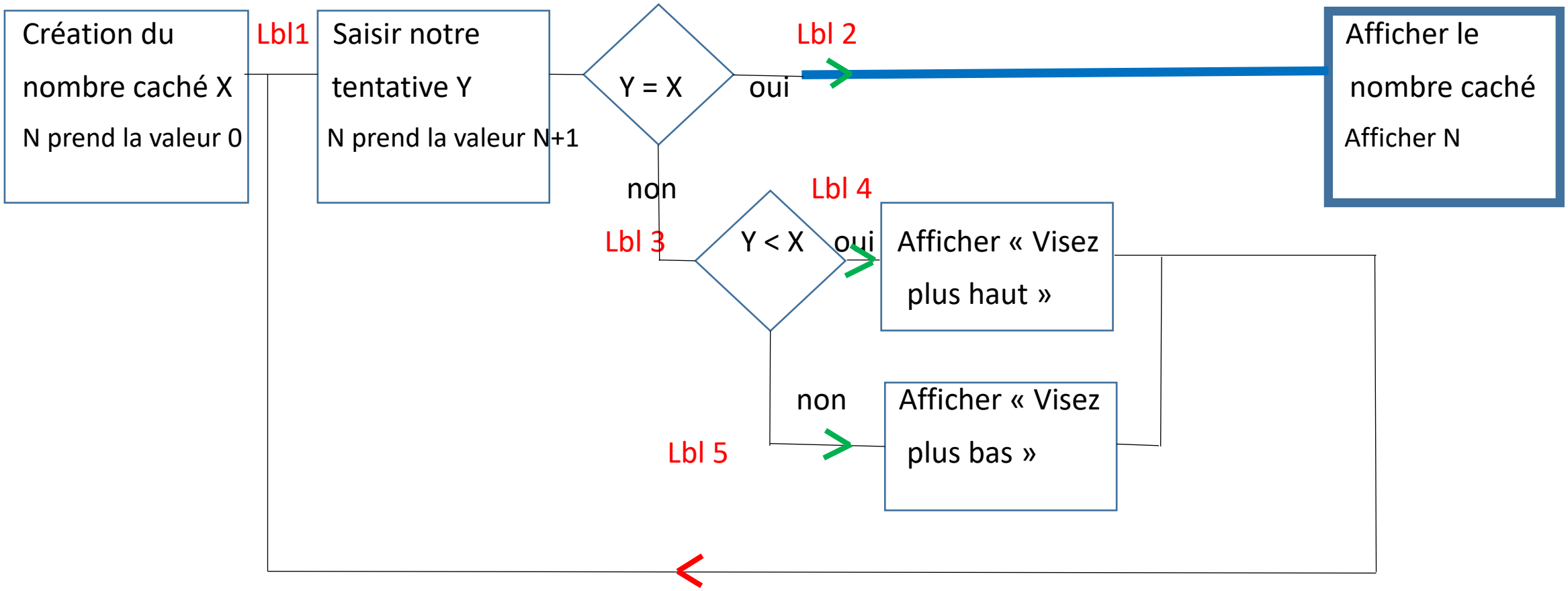
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 : Lbl4 : "+" : Goto1 :



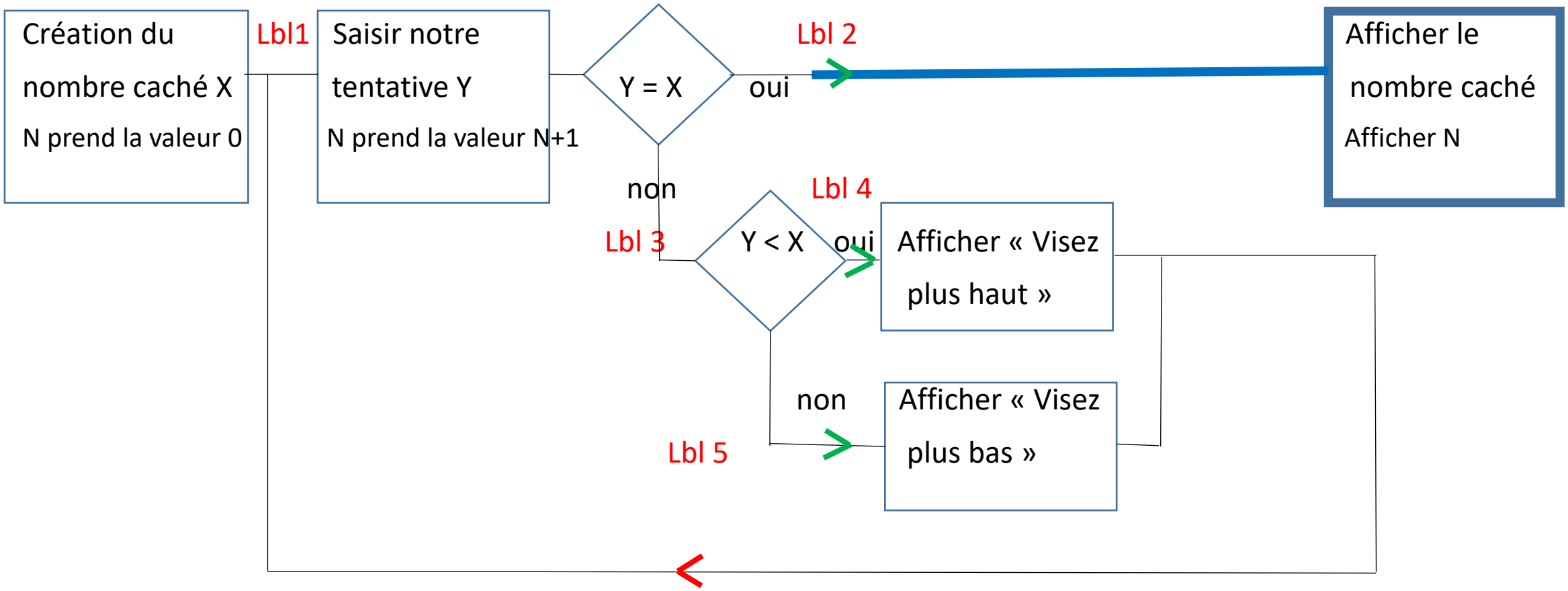
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 : Lbl4 : "+" : Goto1 : Lbl5 : "-" : Goto1 :



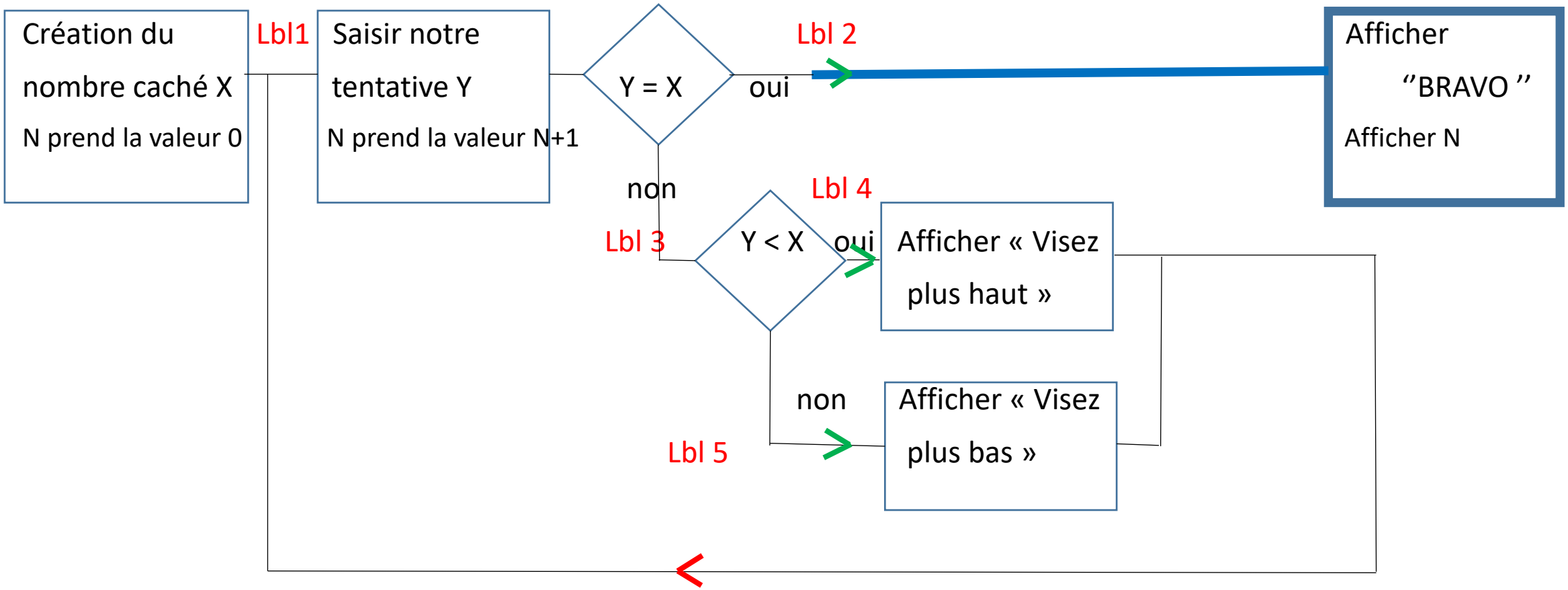
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 : Lbl4 : "+" : Goto1 : **Lbl5 :** "-" : Goto1 :



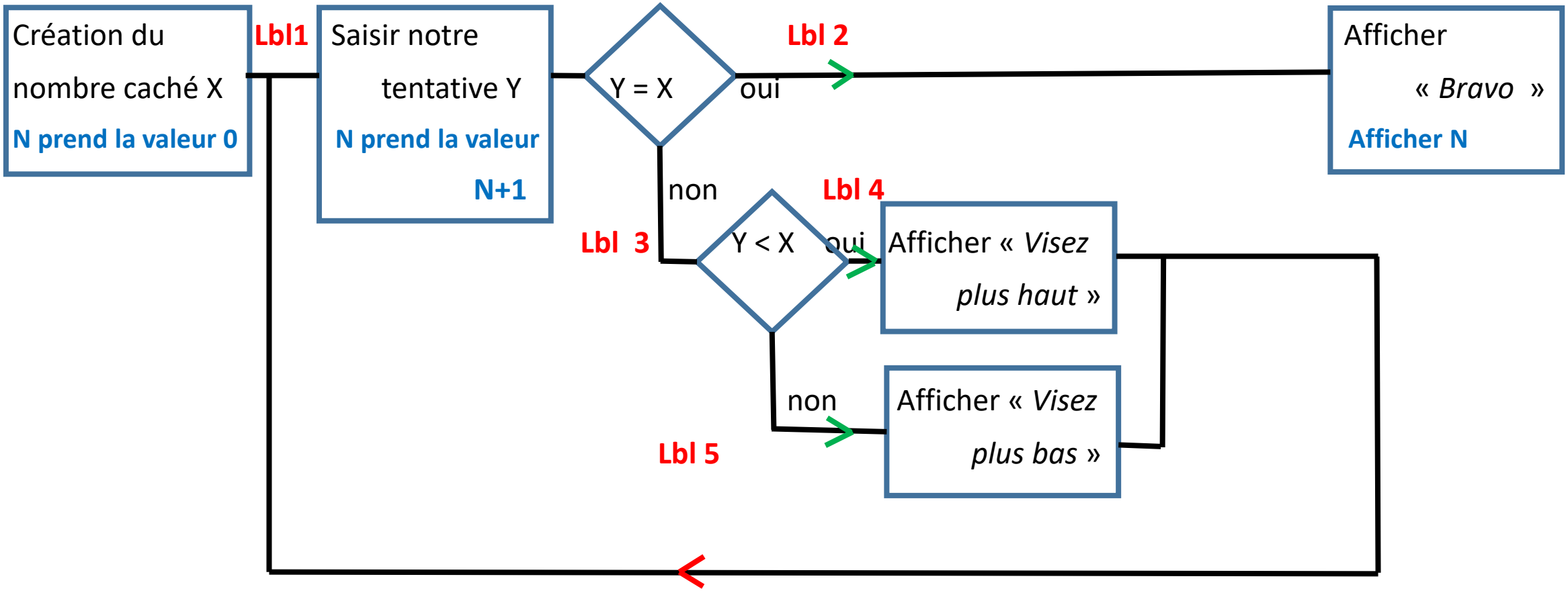
Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 : Lbl4 : "+" : Goto1 : Lbl5 : "-" : Goto1 : : Lbl2 : X◢ N◢



Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 : Lbl4 : ”+” : Goto1 : Lbl5 : ”-” : Goto1 : Lbl2 : ”BRAVO !”: N



Int(1000×RAN#)→X : 0→N : Lbl1 : ?→Y : N+1→N : If Y=X : Then Goto2 : Else Goto3 : Lbl3 : If Y<X : Then Goto4 : Else Goto5 : Lbl4 : ”+” : Goto1 : Lbl5 : ”−” : Goto1 : Lbl2 : ”BRAVO” : N ◿



RAN# se trouve dans OPTN PROBA RAND Int se trouve dans OPTN NUM

Etape 3 : taper le programme. Menu PGM New

Int (1000 × RAN#) → X : **0 → N** : Lbl1 :
? → Y : **N+1 → N** : If Y = X : Then Goto2 :
Else Goto3 : Lbl3 : If Y < X : Then Goto4 :
Else Goto5 : Lbl4 : ”+” : Goto1 : Lbl5 : ”-”
: Goto1 : Lbl2 : ”BRAVO” : **N** ◿

Shift Prgm (: ?◿) puis JUMP (Goto Lbl)

ou COM (If Then Else) ou REL (≥ = <)

OPTN puis NUM (Int) ou PROB (RAND Ran#)

Etape 3 : on tape le programme dans sa calculatrice.

Rappel : (depuis le 1er algorithme !)

Menu → PGRM → NEW → on tape le nom → EXE

→ on tape le programme.

: ?⊿ se trouvent dans Shift Prgm

If Then Else se trouvent dans Shift Prgm → COM

Goto Lbl se trouvent dans Shift Prgm → JUMP

< = se trouvent dans Shift Prgm → REL

RAN# se trouve dans OPTN → PROB → RAND

Int (ou ENT)se trouve dans OPTN → NUM

‘’ et → se trouvent sur le clavier

Etape 4 : on teste le programme en faisant fonctionner sa calculatrice.

et on corrige les erreurs éventuelles :

« Error Syntax » → on appuie sur une flèche directionnelle, la machine nous affiche à quel endroit précis du programme il manque une ponctuation ou une formule mal écrite.

« Error Adress » → on appuie sur une flèche directionnelle, la machine nous affiche à quel endroit précis du programme il manque une adresse Lbl correspondant à un Goto.

Etape 5 : on utilise le programme en faisant fonctionner sa calculatrice.

et on remplit un tableau de valeurs,

ou on joue avec pour cet algorithme,

ou on détermine une réponse, etc…