Répétitions dont le nombre n'est pas connu : la boucle tantque faire

```
tq  c  faire {
    a
ftq
```

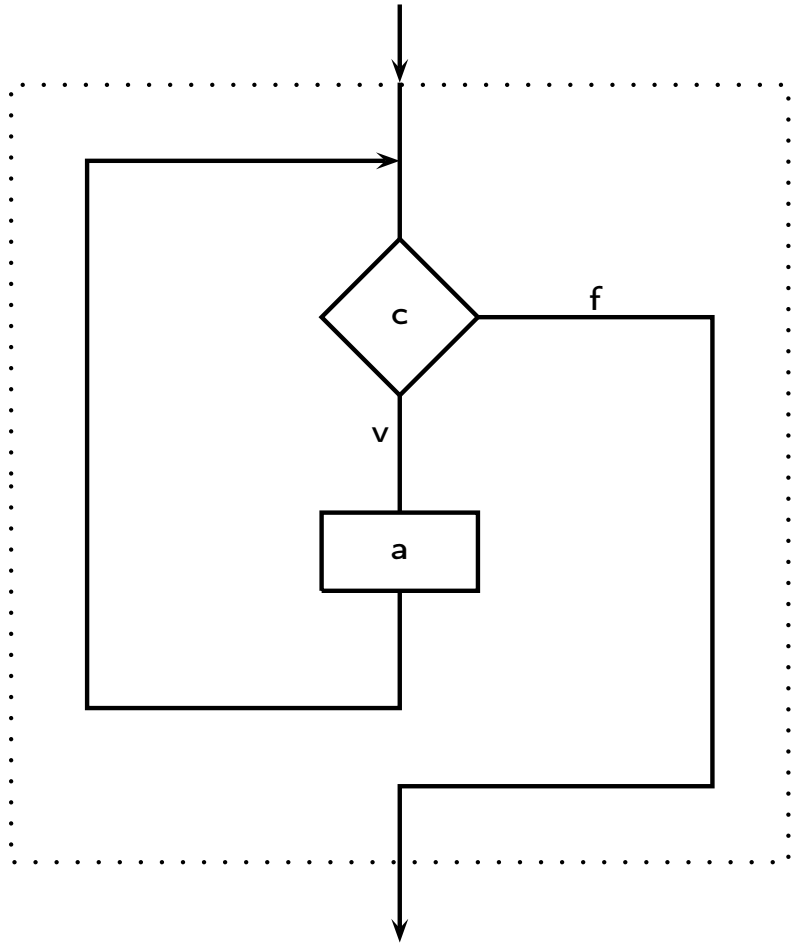
```
while ( c ) {
    a
}
```

Répétitions dont le nombre n'est pas connu : la boucle tantque faire

```
while ( c ) {

    a

}
```



Répétitions dont le nombre n'est pas connu : la boucle tantque faire

- On commence par évaluer la condition **c**
  - si elle est vérifiée, l'action **a** est exécutée, puis on teste à nouveau la condition **c**.
  - si elle n'est pas vérifiée, l'action **a** n'est pas exécutée (on sort du **tantque**)
- La condition **c** doit être définie à l'entrée du **tantque**
- L'action **a** doit modifier la condition **c** pour permettre une sortie du **tantque**.

La boucle tantque faire : Exemple

Calcul de la somme d'une suite de nombres positifs ou nuls saisis à partir du clavier, le dernier nombre est suivi de -1

Par exemple : 10,9,11 et -1.

La boucle tantque faire : Exemple

```
public static int somme ( ) {


}
```

La boucle tantque faire : Exemple

```
public static int somme ( ) {
    int som ;
    som = 0 ;
    int nbre ;
    nbre = EntreeClavier.readInt("nbre␣?␣") ;
    while ( nbre != -1 ) {
        som = som + nbre ;
        nbre = EntreeClavier.readInt("nbre␣?␣") ;
    }
    return som ;
}
```

La boucle tantque faire : Exemple

Ecrire la fonction **saisieNote** qui permet de saisir à partir du clavier un nombre entier compris entre 0 et 20.

Tant que l'utilisateur ne saisit pas un nombre compris entre 0 et 20, un message d'erreur est affiché et on repropose une nouvelle saisie.

Cette fonction n'admet pas de paramètre et retourne la note saisie obligatoirement comprise entre 0 et 20

La boucle tantque faire : Exemple

```
public static int saisirNote ( ) {


}
```

La boucle tantque faire : Exemple

```
public static int saisirNote ( ) {
    int n ;
    n = EntreeClavier.readInt( "note␣?␣" );
    while ( n < 0 || n > 20 ) {
        System.out.print ( "Erreur" ) ;
        System.out.println ( "n␣doit␣être␣compris␣entre␣0␣et␣20" ) ;
        n = EntreeClavier.readInt( "note␣?␣" );
    }
    return n ;
}
```

Recherche séquentielle d'une valeur dans un tableau non rempli

Ecrire la fonction **appartient** qui admet trois paramètres

- **a** un tableau d'entiers dans lequel s'effectue la recherche
- **nbElts** entier égal au nombre d'éléments du tableau précédent
- **valRech** entier valeur recherchée dans le tableau

et qui retourne **vrai** si la valeur recherchée **valRech** appartient au tableau **a** et **faux** dans le cas contaire.

La fonction appartient

```
public static boolean appartient ( int [ ] a , int nbElts , int valRech ) {


}
```

La fonction appartient

```
public static boolean appartient (  int [ ] a , int nbElts , int valRech) {
    boolean trouve ;
    trouve = false ;
    int i ;
    i = 0 ;
    while ( i < nbElts && ! trouve ) {
        if ( a [ i ] == valRech ) {
            trouve = true ;
        } else {
            i = i + 1 ;
        }
    }
    return trouve ;
}
```

Elimination des doublons

Soit le tableau non rempli :

	0	1	2	3	4	5	6	7	8	9	10	11	12
a	10	21	30	30	30	70	21	10	50	50	40	?	?

na	11

Comment obtenir :

	0	1	2	3	4	5	6	7	8	9	10
b	10	21	30	70	50	40	?	?	?	?	?

nb	6

Fonction elimDoublons

```
public static int elimDoublons (  int [ ] a , int na , int [ ] b ) {

    // le tableau b doit etre construit avant l'appel de cette fonction

    // cette fonction retourne nb
}
```

Fonction elimDoublons

```
public static int elimDoublons (  int [ ] a , int na , int [ ] b ) {
    int nb ;
    nb = 0 ;
    for ( int i = 0 ; i < na ; i = i + 1 ) {
        if ( ! appartient ( b , nb , a [ i ] ) {
            b [ nb ] = a [ i ] ;
            nb = nb + 1 ;
        }
    }
    return nb ;
}
```

```
public static void main ( String [ ] args ) {
    int [ ] a ;
    a = new int [ ] { 10,21,30,30,30,70,21,10,50,50,40,0,0} ;
    int na ;
    na = 11 ;
    int [ ] b ;
    b = new int [ na ] ;
    int nb ;
    nb = elimDoublons ( a , na , b ) ;
    for ( int i = 0 ; i < nb ; i = i + 1 ) {
        System.out.print ( b [ i ] + "␣" ) ;
    }
    System.out.println ( ) ;
}
```

L'instruction **for** du langage Java n'est qu'une forme abrégée de l'instruction **while**

```
for ( init ; cond ; incr ) {
    a
}
```

équivaut à

```
init ;
while ( cond ) {
    a
    incr ;
}
```

L'instruction **for** avec un pas **p** positif

```
for ( int i = e1 ; i <= e2 ; i = i + p) {

    a

}
```

équivaut à

```
int i = e1;

while ( i <= e2 ) {
    a

    i = i + p ;
}
```

L'instruction **for** avec un pas -**p** négatif :

```
for ( int i = e1 ; i >= e2 ; i = i - p) {

    a

}
```

équivaut à

```
int i = e1;

while ( i >= e2 ) {
    a

    i = i - p ;
}
```