

LES STRUCTURES CONDITIONNELLES

SI ... ALORS

SI

<expression logique>

ALORS

<liste d'instructions>

FINSI

SI

NombreA > 0

ALORS

ECRIRE (NombreA est un nombre positif)

FINSI

**Si la condition est vraie,
alors la liste d'instructions
est exécutée**

SI ... ALORS ... SINON

SI

<expression logique>

ALORS

<liste d'instructions 1>

SINON

<liste d'instructions 2>

FINSI

Si la condition est **vraie**,
alors la liste d'instructions **1** est exécutée.

Si la condition est **fausse**,
alors la liste d'instructions **2** est exécutée.

SI ... ALORS ... SINON

SI

NbEurosDansLaPoche > 1,70

ALORS

ECRIRE (je peux acheter un ticket de métro)

SINON

ECRIRE (Je vais devoir marcher un peu)

FINSI

ZOOM SUR LA NOTION DE « CONDITION »

Opérateurs de comparaison

- Égalité : SI nombreA = nombreB ALORS ...
- Supériorité :
 - SI nombreA > nombreB ALORS ...
 - SI nombreA >= nombreB ALORS ...
- Infériorité :
 - SI nombreA < nombreB ALORS ...
 - SI nombreA <= nombreB ALORS ...

Conditions multiples

- ET

- SI $nb > 2$ **ET** $nb < 6$ ALORS ...



- Les 2 conditions doivent être remplies pour que la condition finale soit vraie

A	B	A ET B
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

Conditions multiples

- OU

- SI $nb > 2$ **OU** $nb < 6$ ALORS ...


- la condition finale est vraie si **au moins l'une des 2** condition est vraie

A	B	A OU B
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

Conditions multiples

- NON (inversion)
 - SI **NON**(nb < 6) ALORS ...
 - Inverse le résultat de la condition : si c'était faux, ça devient vrai, et si c'était vrai ça devient faux

A	NON(A)
Vrai	Faux
Faux	Vrai

Conditions multiples

- XOR (*OU exclusif*, peu utilisé)
 - Si $\underbrace{\text{nb} > 2}$ XOR $\underbrace{\text{nb} < 6}$ ALORS ...
 - la condition finale est vraie si l'une des 2 conditions est vraie, **mais pas les 2**

A	B	A XOR B
Vrai	Vrai	Faux
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

Exercices

- Exercices sur les structures conditionnelles : 2.1 à 2.3

CONDITIONS IMBRIQUÉES

```
SI (toto est un homme) ET
    (toto est blanc)
ALORS ECRIRE(toto est un homme blanc)
FINSI

SI (toto est une femme) ET
    (toto est blanc)
ALORS ECRIRE(toto est une femme blanche)
FINSI

SI (toto est un homme) ET
    (toto est vert)
ALORS ECRIRE(toto est un homme vert)
FINSI

SI (toto est une femme) ET
    (toto est vert)
ALORS ECRIRE(toto est une femme verte)
FINSI
```

```
SI (toto est un homme)
ALORS
    # On sait ici que toto est un homme
    SI (toto est blanc)
    ALORS
        ECRIRE(toto est un homme blanc)
    SINON
        ECRIRE(toto est un homme vert)
    FINSI
SINON
    # On sait ici que toto est une femme
    SI (toto est blanc)
    ALORS
        ECRIRE(toto est une femme blanche)
    SINON
        ECRIRE(toto est une femme verte)
    FINSI
FINSI
```

```
#!/bin/bash
# Variables : unNombre : nombre réel
# ifImbriques

echo "Entrez un nombre : "
read unNombre

if [ $unNombre -gt 0 ]
then
    echo "nombre positif"
else
    if [ $unNombre == 0 ]
    then
        echo "Zéro"
    else
        echo "nombre négatif"
    fi
fi
```

En python cela donne :

```
num = int(input("Entrez un nombre :"))

if(num>0):
    print("Le nombre est positif")
elif(num<0):
    print("Le nombre est négatif")
else:
    print("Le nombre est égal à zéro")
```


AU CAS OÙ...

AU CAS OÙ <variable> **VAUT**

valeur1 : <liste d'instructions 1>

valeur2 : <liste d'instructions 2>

...

valeurN : <liste d'instructions N>

SINON

<liste d'instructions M>

FIN DE CAS

Cette structure conditionnelle est intéressante dès qu'il y a plus de 2 valeurs possibles, et s'il faut gérer un cas par défaut

```
SI animal = Chien  
ALORS ECRIRE(Ouaf, ouaf)  
FINSI
```

```
SI animal = Chat  
ALORS ECRIRE(Miaou)  
FINSI
```

```
SI animal = Vache  
ALORS ECRIRE(Meuh)  
FINSI
```

...

```
SI animal = Lion  
ALORS ECRIRE(Grrr)  
FINSI
```

```
LIRE (animal)
```

```
AU CAS OÙ (animal) VAUT
```

```
    Chien: ECRIRE (Ouaf, ouaf)
```

```
    Chat: ECRIRE (Miaou)
```

```
    Vache: ECRIRE (Meuh)
```

```
    ...
```

```
    Lion: ECRIRE (Grrr)
```

```
SINON
```

```
    ECRIRE (l'animal est inconnu)
```

```
FIN DE CAS
```

```
#!/bin/bash
# Variables : animal : chaine de caractère
# TestCase

echo "Entrez le nom d'un animal : "
read animal

case $animal in
    Chien) echo "Ouaf, ouaf"
           echo "Ca c'est un bon gros toutou" ;;
    Chat)  echo "Miaou" ;;
    Vache) echo "Meuh" ;;
    Lion)  echo "Grrr" ;;
    *)     echo "l'animal est inconnu" ;;
esac
```

En python cela donne :

```
animal = str(input("Entrez le nom d'un animal : "))
animal.lower()

if(animal == 'chien'):
    print("Ouaf, ouaf. Ca c'est un bon gros toutou")
elif(animal == 'chat'):
    print("Miaou")
elif(animal == 'vache'):
    print("Meuh")
elif(animal == 'lyon'):
    print("Grrrr")
else:
    print("L'animal est inconnu")
```

Exercices

- Exercices sur les structures conditionnelles : 2.4 à 2.9

LES BOUCLES

Les boucles

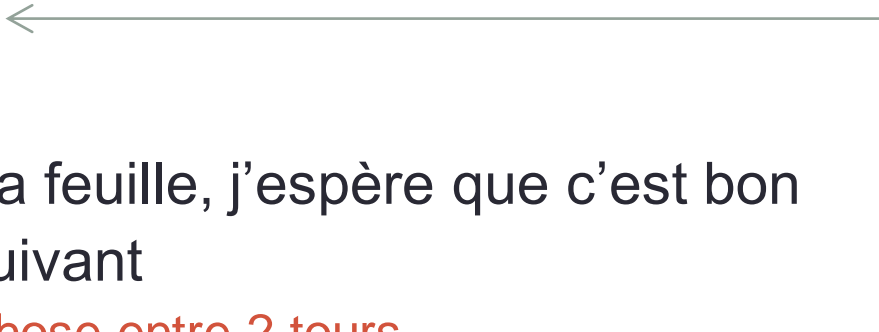
- Répéter un ensemble d'instructions **plusieurs fois**, ou un nombre de fois **illimitée**

Les boucles dans la vie courante

- « Faire le tour des appartements pour voir si quelqu'un a retrouvé mon chien qui a disparu »
- Je commence au premier appart' du couloir
 - → Condition initiale
- Je présente la situation à mon voisin
- Il me répond qu'il ne l'a pas vu
- Je passe à l'appartement suivant
 - → Il se passe quelque chose entre 2 tours
- J'arrête si je trouve mon chien, ou si j'arrive au bout du couloir
 - → Condition finale



Les boucles dans la vie courante

- « Faire les exercices 3 à 6 de la page 315 du bouquin de grec »
 - Je commence à l'exercice 3
 - → Condition initiale
 - Je lis l'énoncé
 - Je me prends la tête
 - J'écris le résultat sur ma feuille, j'espère que c'est bon
 - Je passe à l'exercice suivant
 - → Il se passe quelque chose entre 2 tours
 - J'arrête si j'ai terminé l'exercice 6, ou si j'en ai marre...
 - → Condition finale
- 

POUR... A... PAR PAS DE...

POUR

<variable> \leftarrow Valeur_initiale

A Valeur_finale

PAR PAS DE <pas>

<liste d'instructions>

FINPOUR

POUR... A... PAR PAS DE...

- « Compter de 1 à 10 par pas de 1 »

```
POUR i ← 1 A 10 PAR PAS DE 1  
    ECRIRE (i)  
FINPOUR
```

- Affichage : 1 2 3 4 5 6 7 8 9 10

POUR... A... PAR PAS DE...

- « Compter de 1 à 10 par pas de 2 »

```
POUR i ← 1 A 10 PAR PAS DE 2  
    ECRIRE (i)  
FINPOUR
```

- Affichage : 1 3 5 7 9

Boucle « POUR » en bash

- <http://abs.traduc.org/abs-5.0-fr/ch10.html>

```
#!/bin/bash
```

```
LIMITE=10
```

```
# Double parenthèses, et "LIMITE" sans "$".
```

```
for ((i=1; i <= LIMITE ; i=i+1))
```

```
do
```

```
    echo -n "$i "
```

```
done
```

```
exit 0
```

En python cela donne :

1)

```
for i in range(1,11):  
    print(i)
```

2)

```
for i in range(1,11,2):  
    print(i)
```

3)

```
for i in range(-10,11,2):  
    print(i)
```

TANT QUE... FAIRE

TANT QUE

<condition_pour_rester_dans_la_boucle>

FAIRE

<liste d'instructions>

FINTANTQUE

- La liste d'instruction est exécutée tant que la condition est vraie. Autrement dit, la liste d'instruction va s'exécuter jusqu'à ce que la condition soit fausse.
 - Si la condition est toujours fausse, on ne rentre jamais dans la boucle
 - Si la condition est toujours vraie, **on n'en sort jamais...**

TANTQUE... FAIRE

- « Compter de 1 à 10 par pas de 1 »

```
# Condition initiale
```

```
i ← 1
```

```
TANT QUE i < 10 FAIRE
```

```
    ECRIRE(i)
```

```
    # Définir ce qui se passe entre
```

```
    # chaque tour de boucle
```

```
    i ← i + 1
```

```
FINPOUR
```

- Affichage : 1 2 3 4 5 6 7 8 9 10

TANTQUE... FAIRE

- « Demander à l'utilisateur d'écrire un nombre négatif jusqu'à ce qu'il mette effectivement un nombre négatif »

nombre \leftarrow 0

TANT QUE nombre \geq 0 FAIRE

 ECRIRE (Donnez un nombre négatif)

 LIRE (nombre)

FINPOUR

Boucle « TANTQUE » en bash

```
#!/bin/bash
```

```
i=1
```

```
while [ "$i" -le 10 ]
```

```
do
```

```
    echo -n "$i "
```

```
    i=$(( i+1 ))
```

```
done
```

```
exit 0
```

En python cela donne :

```
1)  n = 1
    while (n<=10):
        print(n)
        n += 1
```

```
2)  m = 10
    p = 0
    while True:
        if(p <= m):
            print(p)
            p += 1
        else:
            break
```

Dans quel cas utilise-t-on quel type de boucle ?

- La boucle TANTQUE peut toujours être utilisée.
- La boucle POUR ne peut servir que dans certains cas :
 - Quand il y a un nombre limité d'itération
 - Quand il se passe quelque chose de simple entre 2 itérations, typiquement $i \leftarrow i + 1$
- Et dans ce cas, la boucle POUR est plus simple à utiliser

Exercices

- Exercices sur les boucles : 3.1 à 3.7