# Conditions et boucles

Alexandre Benoit

BCPST

# Notion d'algorithme

Un algorithme est une procédure permettant de résoudre un problème, écrite de façon suffisamment détaillée pour être suivie sans posséder de compétence particulière ni même comprendre le problème que l'on est en train de résoudre.
Les Instructions de base de l'algorithme sont :

1. La déclaration et l'affectation de variable ;
2. la séquence, qui exécute deux instructions l'une à la suite de l'autre.
3. le test, ou instruction conditionnelle, qui sert à n'exécuter une instruction que dans certains états ;
4. la boucle, qui exécute plusieurs fois la même instruction dans un programme.

# Instructions conditionnelles

Une instruction conditionnelle n'est exécutée que si une condition donnée est vérifiée par l'état courant.

```
if x%2 == 1:
    x=x/2
else:
    x=x+1
```

- Pour identifier sans ambiguïté les instructions appartenant au bloc du if, il est nécessaire de les indenter ;
- if vérifie la valeur d'un booléen. Ici on a le booléen x%2 ;
- la ligne x=x+1 n'est exécuté que si le nombre `x` est pair.
- la ligne x=2*x n'est exécuté que si le nombre `x` est impair.

# Boucle conditionnelle

On se donne les lignes suivantes :

```
p = 1
while c>0:
    p = p*2
    c = c-1
```

- On indente les instructions appartenant au bloc du while comme pour le if ;
- Ce programme bouclera autour des instructions tant que le test ne sera pas vérifié ;
- Ce bloc d'instructions est appelé corps de la boucle et chaque passage est appelé une itération ;
- Cet algorithme sert à calculer $2^c$ ;

# Terminaison d'une boucle

Pour que notre algorithme est un sens, il doit terminer.
Pour montrer que notre boucle termine, on va montrer l'existence d'un variant de boucle.

## Définition

Un variant de boucle est une quantité qui vérifie deux propriétés :

1. Être un entier positif tout au long de l'algorithme ;
2. Décroitre strictement après chaque itération ;

## Exemple

Dans la boucle précédente, l'entier c est un variant de boucle qui prouvera la terminaison.

# Correction d'une boucle

Lorsque la boucle a été écrite, il reste à vérifier qu'elle est correcte, c'est-à-dire qu'elle calcule bien ce que l'on attend.
Une des manières les plus efficaces est d'établir un invariant de boucle.

## Définition

Un invariant de boucle est une propriété qui :

- est vérifiée avant d'entrée dans la boucle ;
- si elle est vérifiée avant une itération, est vérifiée après celle-ci ;
- lorsqu'elle est vérifiée en sortie de boucle permet d'en déduire que le programme est correct.

# Correction de la boucle $2^c$

Pour raisonner, on note $c_i$ et $p_i$ les valeurs des variables $c$ et $p$ après l'exécution de la $i$-*ème* itération. L'état au moment de l'entrée dans la boucle est

$$c_0 = n \qquad p_0 = 1$$

De plus, le corps de la boucle assure les relations suivantes pour toute itération $i$ :

$$c_{i+1} = c_i - 1 \qquad p_{i+1} = 2p_i$$

On remarque alors que la propriété suivante est toujours vérifiée : pour toute itération $i$, on a $p_i = 2^{n-c_i}$ et $c_i \geqslant 0$. En effet :

- Elle est vérifiée pour $i = 0$ : $p_0 = 1 = 2^0 = 2^{n-c_0}$ et $c_0 = n \geqslant 0$.
- Si on a $p_i = 2^{n-c_i}$ et si l'on effectue une itération de plus, on a alors :

$$p_{i+1} = 2p_i = 2^{n-c_i+1} = 2^{n-c_{i+1}} \text{ et } c_{i+1} = c_i - 1.$$

Puisqu'on a effectué l'itération, $c_i > 0$ et donc $c_{i+1} \geqslant 0$.

Toutefois, lorsque la condition $c > 0$ n'est plus vérifiée, en sortie de boucle on a $c_i = 0$ et donc $p_i = 2^n$.

# Boucle for

Lorsque l'on connaît à l'avance le nombre d'itérations qu'il faudra effectuer, on utilise une boucle for

```
p = 1
for c in range(n):
    p = p*2
```

- Ici range(n) représente les nombres entre $0$ et $n-1$ ;
- Cette boucle calcule encore $2^n$.

# Références

Ce cours est issu en grande partie du livre *Informatique pour tous* de Wack et autres aux éditions **Eyrolles**