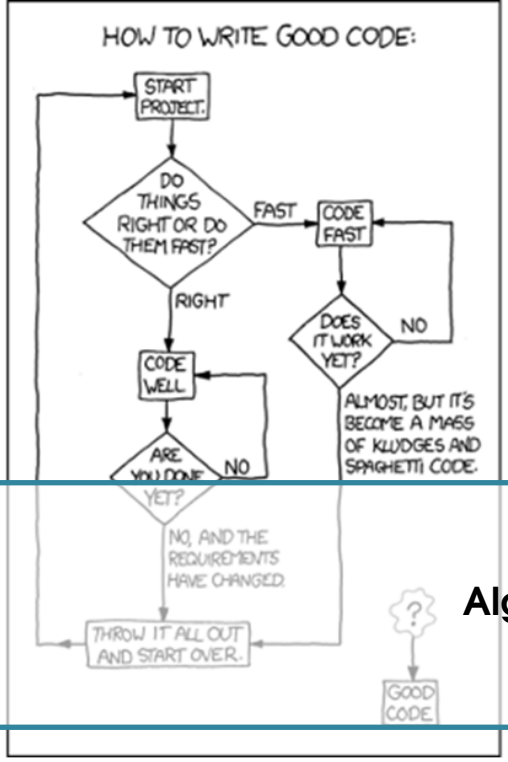

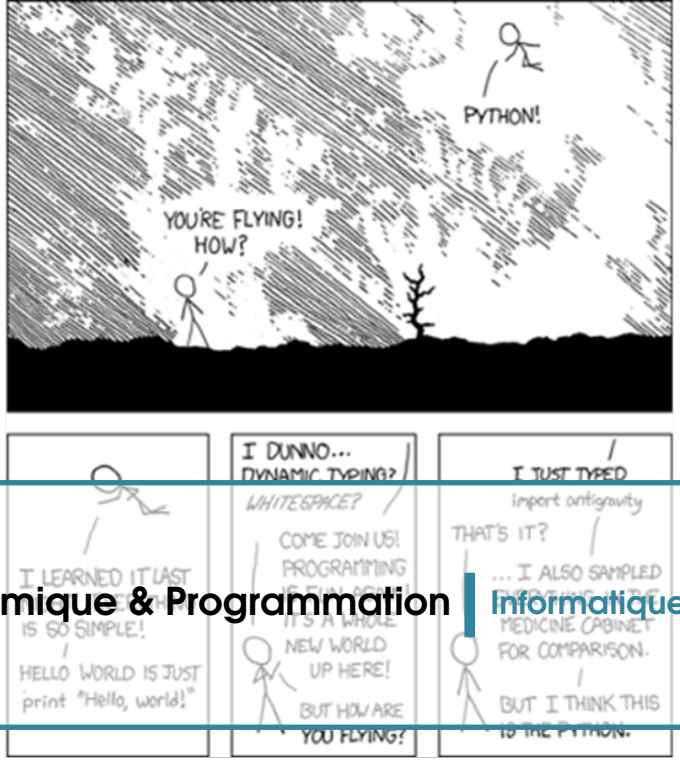

Cours

Chapitre 3
Variant et invariant de boucle

Savoirs et compétences :
- ❒ *justifier qu'une itération (ou boucle) produit l'effet attendu au moyen d'un invariant;*
- ❒ *démontrer qu'une boucle se termine effectivement.*

Xavier Pessoles
PTSI

L'algorithme suivant permet de réaliser une division euclidienne :

Pseudo Code

Data : $a, b \in \mathbb{N}^*$
reste $\leftarrow$ a
quotient $\leftarrow 0$
tant que *reste* $\geq b$ **faire**
    reste $\leftarrow$ reste $- b$
    quotient $\leftarrow$ quotient $+ 1$
fin
Retourner quotient,reste

- Comment s'assurer que l'algorithme en question réalise bien une division euclidienne ?
- Comment s'assurer que la boucle while se termine ?

1 Variant

1.1 Définition

En utilisant l'exemple précédent de la division euclidienne, pour sortir de la boucle, il est nécessaire que le reste devienne, au cours de l'algorithme, inférieur à la valeur de b. Dans ce cas, on constate que b étant un entier naturel, la valeur du reste va diminuer à chaque tour de boucle. Ainsi, on peut intuiter que cet algorithme se terminera.

Définition Variant de boucle

Soit une condition booléenne permettant de sortir d'une boucle constituée d'une comparaison entre une variable et une constante de types entiers positifs. La variable est un variant de boucle si :

- elle reste positive tout au long de l'algorithme ;
- elle décroit (ou croit) strictement à chaque itération de la boucle.

Ainsi, après un nombre fini d'itérations, on est sûr que la boucle se terminera.

R Un variant de boucle **permet** de s'assurer qu'une boucle se terminera.
Un variant de boucle **ne permet pas** de s'assurer qu'un algorithme fournit la réponse attendue.

1.2 Exemple – L'algorithme d'Euclide.

Cet algorithme permet de calculer le PGCD de deux nombres entiers. Il se base sur le fait que si a et b sont deux entiers naturels non nuls, $pgcd(a,b) = pgcd(b, a \bmod b)$.

Pseudo Code

Data : $a, b \in \mathbb{N}^*$
$x \leftarrow a$
$y \leftarrow b$
tant que $y \neq 0$ **faire**
    $r \leftarrow$ reste de la division euclidienne de x par y
    $x \leftarrow y$
    $y \leftarrow r$
fin
Afficher x.

Exemple Calculons le PGCD de 240 et 64 :

$$240 = 3 \cdot 64 + 48$$
$$64 = 1 \cdot 48 + 16$$
$$48 = 3 \cdot 16 + 0$$

■

Dans cet exemple, la condition pour sortir de la boucle est que y devienne nul. y est un variant de boucle si y est un entier positif qui qui décroit strictement à chaque itération de boucle. Dans ces conditions, on est sûr que la boucle se terminera.

Le reste de la division euclidienne de x par y, r, est un entier positif strictement inférieur à y. De plus, à chaque itération, y prend la valeur de r. y décroit donc à chaque itération.

En conséquence, y est un variant de boucle.

2 Invariant de boucle

2.1 Problématique

Lors de la réalisation d'un algorithme, il est nécessaire

- de vérifier que celui-ci permet bien de répondre au problème initial ;
- de s'assurer que celui-ci se termine sans quoi le résultat du problème ne sera jamais délivré à l'utilisateur.

On s'intéresse ici uniquement aux structures itératives. (Les structures récursives seront traitées en seconde année.)

Définition Invariant de boucle [denis]

Un invariant de boucle est une propriété ou une formule logique,
- qui est vérifiée après la phase d'initialisation ;
- qui reste vraie après l'exécution d'une itération ;
- et qui, conjointement à la condition d'arrêt, permet de montrer que le résultat attendu et bien le résultat calculé.

Méthode [**soyeur**]

1. Définir les préconditions (état des variables avant d'entrer dans la boucle).
2. Définir un invariant de boucle.
3. Prouver que l'invariant de boucle est vrai.
4. Montrer la terminaison du programme.
5. Montrer qu'en sortie de boucle, la condition reste vraie.

2.2 Exemple division euclidienne

Utilisons la méthode définie précédemment.

1. Initialement, l'utilisateur saisit les entiers a et b. Le reste prend la valeur de a et me quotient prend la valeur 0.

2. On postule que la quantité $b \cdot \text{quotient} + \text{reste} = a$ est un invariant de boucle.

3. Démontrons par récurrence le postulat précédent.

- Avant d'entrer dans la boucle, on a bien $b \cdot \text{quotient}_0 + \text{reste}_0 = b \cdot 0 + a = a$. Le postulat est donc vrai.
- Admettons le postulat vrai au rang n : $b \cdot \text{quotient}_n + \text{reste}_n = a$.
- Montrons que le postulat est vrai au rang $n+1$:

$$b \cdot \text{quotient}_{n+1} + \text{reste}_{n+1} = b\left(\text{quotient}_n + 1\right) + \text{reste}_n - b = b\,\text{quotient}_n + \text{reste}_n + b - b = a$$

La relation est donc vraie au rang $n+1$.

4. On a montré la terminaison du programme grâce au variant de boucle.

5. En sortie de programme, l'invariant de boucle reste vérifié.

2.3 Exemple – L'algorithme d'Euclide.

Cet algorithme permet de calculer le PGCD de deux nombres entiers. Il se base sur le fait que si a et b sont deux entiers naturels non nuls, $pgcd(a,b) = pgcd(b, a \bmod b)$.

▪ **Pseudo Code**

Data : $a, b \in \mathbb{N}^*$
$x \leftarrow a$
$y \leftarrow b$
tant que $y \neq 0$ **faire**
  $r \leftarrow$ reste de la division euclidienne de x par y
  $x \leftarrow y$
  $y \leftarrow r$
fin
Afficher x.

▪ **Exemple** Calculons le PGCD de 240 et 64 :

$$\begin{aligned} 240 &= 3 \cdot 64 + 48 \\ 64 &= 1 \cdot 48 + 16 \\ 48 &= 3 \cdot 16 + 0 \end{aligned}$$

$pgcd(240, 64)$ est le dernier reste non nul à savoir 16.

▪

Démonstration Soient a et b, tels que $(a;b)\in\mathbb{N}^*$ tels que $a>b$. Soient $(q;r)\in\mathbb{N}$ tels que $a=b\cdot q+r$. Avec r le reste de la division euclidienne de a par b et q le quotient.

Montrons que $PGCD(a,b)=PGCD(b,r)$**.**

Soit $d\in\mathbb{N}$ un diviseur commun de a et b. Dans ce cas, d divise $a-b\cdot q$; donc d divise r.

En conclusion, si d divise a et b alors $PGCD(a,b)=PGCD(b,r)$.

Réciproquement, soit $d\in\mathbb{N}$ un diviseur commun de b et r. Dans ce cas, d divise $b\cdot q+r$; donc d divise a.

En conclusion, si d divise b et r alors $PGCD(a,b)=PGCD(b,r)$.

On a donc $PGCD(a,b)=PGCD(b,r)$.

Démonstration **Preuve par invariant de boucle de l'algorithme d'Euclide**

1. Initialement, a et b sont des données saisies par l'utilisateur. On a $x_0=a$ et $y_0=b$.
2. On postule que l'invariant de boucle est donné par $x=y\cdot q+r\Longleftrightarrow r=x-y\cdot q$.
3. - Lors de la première itération, r_1 est tel que $r_1=x_1-y_1\cdot q_1$.
   - Lors de la nième itération, r_n est tel que $r_n=x_n-y_n\cdot q_n$.
   - À l'itération suivante, $x_{n+1}=y_n$ et $y_{n+1}=r_n$, on peut donc trouver r_{n+1} tel que $r_{n+1}=x_{n+1}-y_{n+1}\cdot q_{n+1}$.
4. On a montré précédemment que l'itération se terminait.
5. En sortant de la boucle, l'invariant reste vrai.

2.4 Exemple

▪ **Exemple** Écrire la fonction qui permet de calculer la valeur prise par 2^n. On utilisera pour cela une boucle *while* qui sera mise en œuvre à l'aide d'un variant et d'un invariant de boucle qu'il faudra définir. ▪

Intuitivement, le premier algorithme que l'on peut proposer est le suivant :

▪ **Python**
```
def fonction(n):
    i=0
    res=1
    while i<n:
        res = res*2
        i=i+1
    return res
```

Prenons quelques exemples :
- pour $n=0$: on retourne directement 1 ;
- pour $n=1$: on retourne 2 *etc.*

Pour mettre en place le variant, on va commencer par mettre en place un variant qui doit décroître :

▪ **Python**
```
def fonction(p):
    i=p
    res=1
    j = 0
    while i!=0:
        res = res*2
        i=i-1
        j = j+1
    return res
```

i est donc un variant de boucle car :
- i est toujours un entier positif ;
- i décroît strictement à chaque itération.

Proposons l'invariant de boucle suivant : $\text{res}_j=2^{p-i_n}$

L'algorithme permet d'écrire les suites suivantes :
- $i_n=i_{n-1}-1$ avec $i_0=p$;
- $res_n=2\cdot res_{n-1}$ avec $res_0=p$

Avant d'entrée dans la boucle, au rang $j=0$: $i_0=p$, $res_0=1$. On a donc $res_0=2^{p-i_0}=2^0=1$.

On suppose la propriété vraie au rang $j=n$ et donc que $\text{res}_j=2^{p-i_n}$.

Au rang $j = n+1$, on a : $i_{n+1} = i_n + 1$ et

$$res_{n+1} = 2 \cdot 2^{p-i_n} =$$

.

On a donc $res_{n+1} = 2 \cdot res_n = 2 \cdot 2^{p-i_n} = 2^{p-i_n+1} = 2^{p-i_{n+1}}$ (CQFD).
La propriété est donc vraie au rang $n+1$.
Vérifions que la propriété est vraie en fin de boucle.
La boucle s'exécute p fois. En conséquence, en fin de boucle, $i_n = 0$ et $j = n$. En conséquence, $res_p = 2^{p-0} = 2^p$.

Références

[1] François Denis `http://pageperso.lif.univ-mrs.fr/~francois.denis/algoL2/chap1.pdf`

[2] Alain Soyeur `http://asoyeur.free.fr/`

[3] Wack et Al., *L'informatique pour tous en classes préparatoires aux grandes écoles*, Editions Eyrolles.

[4] Yves Benjamin, Vérification de la validité d'un programme.