R1.01 : Initiation au développement Cours 2

M.Adam, N.Delomez, JF.Kamp, L.Naert

IUT de Vannes

4 août 2022

Sommaire

Calcul du PGCD

Le PGCD, Plus Grand Commun Diviseur, de 50 et 125.

Algorithme de calcul du PGCD

Le PGCD de deux nombres est obtenu en soustrayant le plus petit des deux nombres au plus grand jusqu'à ce que les deux soient égaux.

q	p
50	125
50	75
50	25
25	25

- Sous JavaTutor : https://tinyurl.com/C02PJPGCD
- Sous AlgoTouch : https://tinyurl.com/C02AGTPGCD

Il nous manque une structure de contrôle pour permettre de réaliser cet algorithme : la boucle.

La boucle ou l'itération

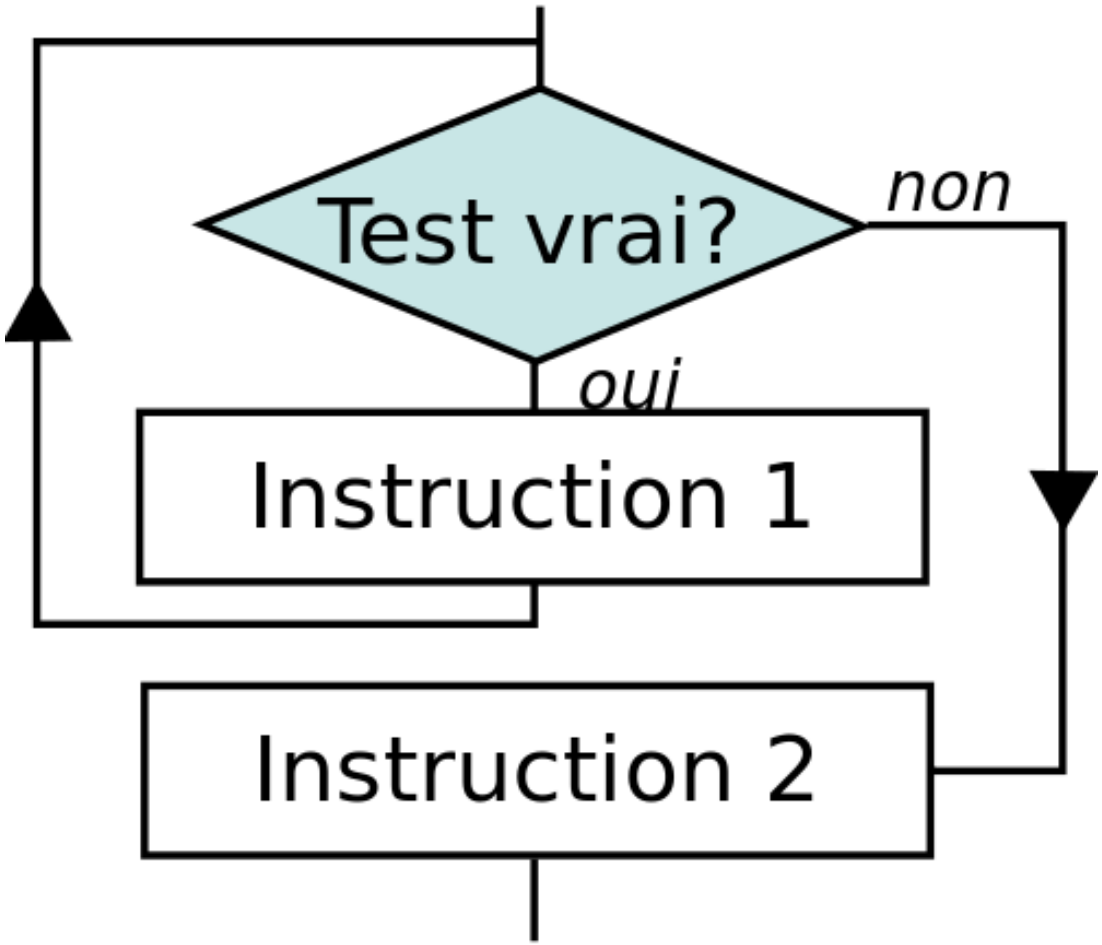
https ://tinyurl.com/C02AGTPGCD
Syntaxe :

```
while (condition) {
    instructions
}
```

Interprétation d'une boucle

Répéter les `instructions` tant que la `condition`, `Test`, est vraie.

```
while (Test) {
    instruction 1;
}
instruction 2;
```



Premier exemple

Le programme soustrait 1 à la variable `i` jusqu'à ce qu'elle soit égale à 0.

```
/**
* décrémente i jusqu'à 0
* @author M.Adam
*/
class Dec {
    void principal () {
        short i ;
        i = (short) SimpleInput.getInt ("Valeur de i = ");
        while (i != 0) {
            System.out.println ("Valeur de i : " + i);
            i = (short) (i - 1);
        }
        System.out.println ("Valeur de i : " + i);
    }
}
```

```
$ java -cp ../class Start "Dec"
Valeur de i = 3
Valeur de i : 3
Valeur de i : 2
Valeur de i : 1
Valeur de i : 0
$
```

- Sous JavaTutor : https://tinyurl.com/C02JTDec
- Sous AlgoTouch : https://tinyurl.com/C02AGTDec

Instructions	i	écran et clavier
`i = SimpleInput.getInt ("Valeur de i = ");`	3	3
`while (i != 0) {`	3	3
`i = i - 1;`	2	
`while (i != 0) {`	2	2
`i = i - 1;`	1	
`while (i != 0) {`	1	1
`i = i - 1;`	0	
`while (i != 0) {`	0	
`System.out.println ("Valeur de i : " + i);`	0	0

Les différentes parties d'une boucle

- L'initialisation

```
i = SimpleInput.getInt ("Valeur de i = ");
```

- La condition de continuation :

```
i != 0
```

- La condition d'arrêt :

`!(i != 0)` qui est équivalent à `i == 0`

- Le corps de boucle :

```
System.out.println ("Valeur de i : " + i);
i = i - 1;
```

- La terminaison :

```
System.out.println ("Valeur de i : " + i);
```

Autres formes de boucles

Il existe d'autres boucles :

- `do ... while`
- `for`

Ce ne sont que d'autres formes de la boucle `while`.

Les restrictions appliquées sur cette séquence

- Une seule condition de terminaison
- Pas de boucles imbriquées

Sommaire

Les risques d'erreurs

- La boucle infinie,
- l'erreur de calcul.

Boucle infinie

- Il faut que le corps de la boucle rende la condition de continuation fausse,
- mais ce n'est pas évident !

Exemple 1 : Boucle infinie ou pas ?

```
/**
* décrémente i jusqu'à 0
* @author M.Adam
*/
class Dec {
    void principal () {
        short i ;
        i = (short) SimpleInput.getInt ("Valeur de i = ");
        while (i != 0) {
            System.out.println ("Valeur de i : " + i);
            i = (short) (i - 1);
        }
        System.out.println ("Valeur de i : " + i);
    }
}
```

Instructions	i	écran et clavier
`i = SimpleInput.getInt ("Valeur de i")`		
`while (i != 0) {`		
`i = i - 1;`		

Exemple 2 : Boucle infinie ou pas ?

```
while (i != j) {
  if (i < j) {
    i = i + 1;
    j = j - 1;
  } else {
    i = i - 1;
    j = j + 1;
  }
}
```

Exemple 3 : Boucle infinie ou pas ?

```
while (i != 0) {
  if (i < 0) {
      i = i + 1;
  } else {
      i = i - 1;
  }
}
```

Exemple 4 : Boucle infinie ou pas ?

```
rep = SimpleInput.getChar ("rep : ");
while (rep != 'o' || rep != 'n') {
    rep = SimpleInput.getChar ("rep : ");
}
```

La condition de sortie est :

Pas facile de gérer les conditions de continuation avec plusieurs parties disjonctives, `||`, ou conjonctives, `&&` !

L'erreur de calcul

Le résultat rendu pas la boucle n'est pas celui attendu.

Exemple 1 : Correct ou pas ?

```
/**
* Calcul de a^n
* @author M.Adam
*/
class Exposant {
    void principal () {
        float a, exp;
        int  n, i;
        boolean negatif;

        System.out.println ("Calcul de a^n");
        a = SimpleInput.getFloat ("Valeur de a = ");
        n = SimpleInput.getInt   ("Valeur de n = ");
        negatif = false;
        if (n < 0) {
            negatif = true;
            n = (-1) * n;
        }
        exp = 1;
        i = 0;
        while (i != n) {
            exp = exp * a;
            i = i + 1;
        }
        if (negatif) {
            exp = 1 / exp;
        }
        System.out.println ("Valeur de a^n : " + exp);
    }
}
```

Exemple 2 : Correct ou pas ?

La formule de formule de Gregory-Leibniz permet de calculer pi :

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ... + \frac{(-1)^n}{(2n+1)}$$

```
4
2.6666665
3.4666665
2.895238
3.3396823
2.976046
2.976046
```

$$\frac{\pi}{4} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + \frac{(-1)^n}{(2n+1)}$$

```
/**
* calcul d'une valeur approchée de pi
* @author M.Adam
*/
class Pi {
    void principal () {
        float pi = 1;
        int i = 0;
        int q = 3;
        int signe = -1;
        while (i < 100) {
            pi = pi + (signe / q);
            System.out.println ("Valeur de Pi : " + pi * 4);
            signe = (-1) * signe;
            q = q + 2;
            i = i + 1;
        }
        System.out.println ("Valeur de Pi : " + pi * 4);
    }
}
```

Exemple 3 : Correct ou pas ?

Nous allons résoudre par essais successifs l'équation $x^3 + x^2 = 100$.

- La fonction f est définie par $f(x) = x^3 + x^2$ sur $[0\,;\,10]$.
- Cette fonction est croissante sur $[0\,;\,10]$ et on peut calculer que : $f(4) = 80$ et $f(5) = 150$.
- Par conséquent on peut trouver une solution approximative de l'équation $x^3 + x^2 = 100$ sur $[4\,;\,5]$. Nous allons donc procéder de la manière suivante :
  - calculer $f(4)$
  - puis calculer $f(4.01)\,;\,f(4.02)\,;\,f(4.03)$ et ainsi de suite, et de faire continuer l'algorithme tant que l'image de chaque nombre est inférieure à 100.

```
/**
* Calcul de x*x*x+x*x=100 par approximation
* @author M.Adam
*/
class Approximation {
    void principal () {
        double x;
        double f;

        x = 4;
        f = x * x * x + x * x;

        while (f < 100) {
            x = x + 0.01;
            f = x * x * x + x * x;
        }
        System.out.println ("f(" + x + ") = " + f);
    }
}
```

Sommaire

En résumé

- La boucle `while`
- Les risques d'erreurs

A venir

- La construction des boucles
- La structure de tableau