

Éléments de programmation : boucles

Valeur d'accueil et de reconversion en informatique (VARI1)

Daniel Porumbel (dp.cnam@gmail.com)

<http://cedric.cnam.fr/~porumbed/vari1/>

Introduction

Soit le programme à gauche

- observer les instructions **if** répétitives
- comment automatiser ces instructions ?

Sans boucle

```
void setup() {  
    int [] tab = new int [4];  
    tab [0]=5; tab [1]=4;  
    tab [2]=2; tab [3]=1;  
    int min=tab [0];  
    if (tab [1]<min)  
        min = tab [1];  
    if (tab [2]<min)  
        min = tab [2];  
    if (tab [3]<min)  
        min = tab [3];  
    println (min);  
}
```

Introduction

Soit le programme à gauche

- observer les instructions **if** répétitives
- comment automatiser ces instructions ? **utiliser la boucle for**

Sans boucle

```
void setup() {  
    int [] tab = new int [4];  
    tab [0]=5; tab [1]=4;  
    tab [2]=2; tab [3]=1;  
    int min=tab [0];  
    if (tab [1]<min) }i= 1  
        min = tab [1];  
    if (tab [2]<min) }i= 2  
        min = tab [2];  
    if (tab [3]<min) }i= 3  
        min = tab [3];  
    println (min);  
}
```

Introduction

Soit le programme à gauche

- observer les instructions **if** répétitives
- comment automatiser ces instructions ? **utiliser la boucle for**

Sans boucle

```
void setup() {  
  int [] tab = new int [4];  
  tab [0]=5; tab [1]=4;  
  tab [2]=2; tab [3]=1;  
  int min=tab [0];  
  if (tab [1]<min) } i= 1  
    min = tab [1];  
  if (tab [2]<min) } i= 2  
    min = tab [2];  
  if (tab [3]<min) } i= 3  
    min = tab [3];  
  println (min);  
}
```

Boucle For

```
void setup() {  
  int [] tab = new int [4];  
  tab [0]=5; tab [1]=4;  
  tab [2]=2; tab [3]=1;  
  int min=tab [0];  
  int i;  
  for (i=1; i<4; i++){  
    if (tab [i]<min)  
      min = tab [i];  
  }  
  println (min);  
}
```

Introduction

Soit le programme à gauche

- observer les instructions **if** répétitives
- comment automatiser ces instructions ?

Sans boucle

```
void setup() {  
  int [] tab = new int [4]  
  tab [0]=5; tab [1]=4;  
  tab [2]=2; tab [3]=1;  
  int min=tab [0];  
  if (tab [1]<min) {  
    min = tab [1]; } i= 1  
  if (tab [2]<min) {  
    min = tab [2]; } i= 2  
  if (tab [3]<min) {  
    min = tab [3]; } i= 3  
  println (min);  
}
```

Boucle For

```
void setup() {  
  int [] tab = new int [4]  
  tab [0]=5; tab [1]=4;  
  tab [2]=2; tab [3]=1;  
  int min=tab [0];  
  int i;  
  for (i=1; i<4; i++) {  
    if (tab [i]<min)  
      min = tab [i];  
  }  
  println (min);  
}
```

condition pour continuer

initialisation i

Incrémentation avant l'itération suivante

Avantages boucles : le passage à l'échelle, trouver le minimum d'un tableau de 1000 valeurs

```
void setup() {  
    int[] tab = new int[1000];  
    int i;  
    //initialisation tableau  
    for (i=0; i<1000; i++) //une seule instruction=>  
        tab[i]= i;        //pas besoin d'accolades  
    //calcul minimum  
    int min=tab[0];  
    for (i=1; i<1000; i++){  
        if (tab[i]<min)  
            min = tab[i];  
    }  
    println (min);  
}
```

Une fonction avec une boucle

```
float calcMin(float [] t){
    float min=t[0];
    for (int i=1; i<1000; i++){ //i déclaré dans le
        if (t[i]<min)           //1er champs du for
            min = t[i];
    }
    return min;
}

void setup() {
    float [] tab = new float[1000];
    int i;
    for (i=0; i<1000; i++)      //tableau avec des
        tab[i]= random(999999); //vals aléatoires
    float m = calcMin(tab);
    println(m);
}
```

Quel est le résultat des deux programmes ?

```
1 println ( "*****" );  
2 println ( "*****" );  
3 println ( "*****" );  
4 println ( "*****" );  
5 println ( "*****" );
```

```
1 int i;  
2 for (i=1; i<=5; i++)  
3     println ( "*****" );
```


Quel est le résultat du code ?

```
void setup () {  
  size(400,400);  
  int i;  
  for (i=0; i<15; i++){  
    // fill(10*i,0,0);  
    fill(i*20,0,0);  
    ellipse(i*10,i*20,10*i, 10*i);  
  }  
}
```

Boucles `for` et boucles `while`

- La plus classique boucle `for`

```
for (int i=0; i<100; i++){  
    ... //faire quelque chose (100 fois)  
}
```

- La plus classique boucle `while` (tant que)

```
int i = 0;  
while (i<100){  
    ... //faire quelque chose  
    ... //100 fois si la seule ligne qui  
    ... //modifie i est la suivante  
    i = i+1;  
}
```

- `for (init; cond; incr)` s'exécute tant que `cond` est vraie
 - Exemple : `for (int i=0; i<10; i++)` s'exécute tant que `i<10` est vrai
- `while (cond)` s'exécute tant que `cond` est vraie

Boucle for

```
void setup() {
  int[] tab = new int[1000];
  for (int i=0; i<1000; i++)
    tab[i] = i;
  // calcul du minimum
  int min = tab[0];
  int i = 1;
  for (i = 1; i < 1000; i++) {
    if (tab[i] < min)
      min = tab[i];
  }
  println(min);
}
```

- `for (init; cond; incr)` s'exécute tant que `cond` est vraie
 - Exemple : `for (int i=0; i<10; i++)` s'exécute tant que `i<10` est vrai
- `while (cond)` s'exécute tant que `cond` est vraie

Boucle for

```
void setup () {  
    int [] tab = new int [1000];  
    for (int i=0; i<1000; i++)  
        tab [i] = i;  
    // calcul du minimum  
    int min = tab [0];  
    int i = 1;  
    for (i = 1; i < 1000; i++) {  
        if (tab [i] < min)  
            min = tab [i];  
    }  
    println (min);  
}
```

- `for(init;cond;incr)` s'exécute tant que `cond` est vraie
 - Exemple : `for(int i=0;i<10;i++)` s'exécute tant que `i<10` est vrai
- `while(cond)` s'exécute tant que `cond` est vraie

Boucle for

```
void setup() {
  int[] tab = new int[1000];
  for(int i=0;i<1000;i++)
    tab[i]= i;
  //calcul du minimum
  int min=tab[0];
  int i=1;
  for(i=1;i<1000;i++){
    if(tab[i]<min)
      min = tab[i];
  }
  println(min);
}
```

Boucle while

```
void setup() {
  int[] tab = new int[1000];
  for(int i=0;i<1000;i++)
    tab[i]= i;
  //calcul du minimum
  int min=tab[0];
  int i=1;
  while(i<1000){
    if(tab[i]<min)
      min = tab[i];
    i ++ ; //équivalent i = i+1
  }
  println(min);
}
```

Rappel `while`

```
while (condition) {  
    ...  
}
```

- 1 Écrire une fonction qui trace des lignes le long des coordonnées $(x[0], y[0])$, $(x[1], y[1])$, $(x[2], y[2])$, $(x[3], y[3])$,
 - La taille d'un tableau : `x.length`
- 2 Écrire une boucle `while` pour initialiser un tableau de `floats` avec 50 valeurs aléatoires (utiliser `random`)
- 3 Appliquer la fonction du point 1 sur des tableaux aléatoires

Rappel `while`

```
while (condition) {  
    ...  
}
```

- 1 Écrire une fonction qui trace des lignes le long des coordonnées $(x[0], y[0])$, $(x[1], y[1])$, $(x[2], y[2])$, $(x[3], y[3])$,
 - La taille d'un tableau : `x.length`
- 2 Écrire une boucle `while` pour initialiser un tableau de `floats` avec 50 valeurs aléatoires (utiliser `random`)
- 3 Appliquer la fonction du point 1 sur des tableaux aléatoires

Rappel `while`

```
while (condition) {  
    ...  
}
```

- 1 Écrire une fonction qui trace des lignes le long des coordonnées $(x[0], y[0])$, $(x[1], y[1])$, $(x[2], y[2])$, $(x[3], y[3])$,
 - La taille d'un tableau : `x.length`
- 2 Écrire une boucle `while` pour initialiser un tableau de `floats` avec 50 valeurs aléatoires (utiliser `random`)
- 3 Appliquer la fonction du point 1 sur des tableaux aléatoires