Introduction

Un ordinateur est fait pour effectuer des calculs longs et répétitifs.

Dans ce chapitre, nous introduisons une nouvelle instruction, la boucle, qui permet d'exécuter une instruction plusieurs fois. Nous en présentons deux variantes, la boucle for et la boucle while. Nous expliquons comment manipuler le compteur d'une boucle for, dans quels cas plutôt utiliser une boucle while, et pourquoi il peut arriver que l'exécution d'une boucle ne s'arrête jamais.

Gilles Kahn (1946-2006) et Gordon Plotkin (1946-) ont proposé des outils pour décrire la sémantique des langages de programmation, c'est-à- dire ce qu'il se passe quand on exécute un programme. Gilles Kahn est aussi l'auteur, avec Gérard Huet, du système Mentor, l'un des premiers systèmes qui permet de définir de manière complètement formelle des langages de programmation. On doit à Gordon Plotkin des contributions à de nombreux domaines de l'informatique, en particulier la démonstration automatique et la théorie des systèmes concurrents.

La boucle for

Un ordinateur est fait pour effectuer des calculs longs et répétitifs.

Exemple, exécuter la boucle :

```
for (i = 1; i <= 10; i = i + 1)
{
      print("allô ");
}
println("t'es où ?");
```

a pour effet d'afficher :

allô allô allô allô allô allô allô allô allô allô t'es où ?

La boucle for

Exercice 1

Ecrire un programme qui affiche les 31 jours du mois de Janvier à l'aide d'une boucle for.

Correction exercice 1

```
void main(){
        int jour;
        for (jour=1;jour<=31;jour=jour+1)
        {
                print(jour);
                println(" Janvier");
        }
}
```

La boucle for

Exercice 2

Écrire un programme qui affiche un calendrier pour une année entière.

Correction exercice 2

```
void main() {
   int jour,mois,nb_jours;
   for (mois = 1; mois <= 12; mois = mois + 1) {
      if (mois == 2) nb_jours = 28;
      else if (mois == 1 || mois == 3 || mois == 5 || mois == 7 || mois == 8
|| mois == 10 || mois == 12) nb_jours = 31;
      else nb_jours = 30;
      for (jour = 1; jour <= nb_jours; jour = jour + 1) {
         println(jour+"/"+mois);
      }
      println("");
   }
}
```

La boucle for

Exercice 3

La suite de nombres définie par récurrence de la manière suivante :

- $u_0 = 13$
- $u_{n+1} = (16805\ u_n + 1)\ \%\ 32768$ semble aléatoire.

a) Écrire un programme qui affiche les 10 000 premiers termes de cette suite.

b) Pour simuler une suite de tirages à pile ou face, on observe le neuvième bit de chaque élément de cette suite et on décrète, lors du *i*-ème tirage, que la pièce est tombée du côté pile si le neuvième bit du nombre u_i est un 0, et qu'elle est tombée du côté face si c'est un 1.
Écrire un programme qui affiche les 10 000 premiers tirages.

c) On teste la qualité de ce générateur d'aléa en comptant le nombre de fois que la pièce tombe d'un côté et de l'autre.
Écrire un programme qui simule 10 000 tirages et compte le nombre de fois que la pièce tombe du côté pile.

d) Qu'obtient-on si on observe le bit des unités au lieu d'observer le neuvième bit ?
Expliquer pourquoi : montrer que si u_n est pair alors u_{n+1} est impair et que si u_n est impair alors u_{n+1} est pair.

e) Montrer que, loin d'être réellement aléatoire, la suite u est en fait périodique à partir d'un certain rang.

.

La boucle for

Correction exercice 3

a)

```
void main() {
   long u;
   u = 13;
   for (int i = 0;i<=1000;i++)
   {
        println("u("+i+") = " + u);
        u = (16805*u  + 1)% 32768;
   }
}
```

b)

```
void main() {
   long u;
   u = 13;
   for (int i = 0;i<=1000;i++)
   {
            if ((u & 256)  < 256)
                     println("Pile");
                     else
                     println("Face");
            u = (16805*u  + 1)% 32768;
   }
}
```

La boucle for

Correction exercice 3

c)

```
void main() {
   int u;
   u = 13;
   int nb_piles;
   nb_piles = 0;
   for (int i = 0; i <= 100000; i ++) {
      if ((u & 256) < 256) nb_piles ++;
      u = (16805 * u + 1) % 32768;
   }
   println("Nombre de piles obtenus : " +
nb_piles);
}
```

Le programme affiche 49 993 piles
Ce qui est très proche d'une situation équirépartie (50 000)

La boucle for

Correction exercice 3

d)

```
void main() {
   long u;
   u = 13;
   for (int i = 0;i<=1000;i++)
   {
            if ((u & 1)  < 1)
                     println("Pile");
                     else
                     println("Face");
            u = (16805*u  + 1)% 32768;
   }
}
```

Le programme affiche en alternance Face, Pile, Face, Pile, ….

Si u_n est pair alors $16805u_n + 1$ sera impair
De même que le reste de la division euclidienne de ce nombre par 32768
Donc u_n est impair.

Si u_n est impair alors $16805u_n + 1$ sera pair
De même que le reste de la division euclidienne de ce nombre par 32768
Donc u_{n+1} est pair.

La boucle for

Correction exercice 3

e) On peut se servir du programme suivant pour montrer que $u_{32678} = u_0 = 13$
On a donc $u_{n + 32768} = u_n$

```
void main() {
   int u;
   u = 13;
   int[] tab_u;
   tab_u = new int[100001];
for (int i = 0; i <= 100000; i ++) {
      for (int j = 0; j < i; j ++) {
         if (tab_u[j] == u) {
             println("u = " + u + " i = " + i + " j = " + j);
            return ;
         }
      }
      tab_u[i] = u;
      u = (16805 * u + 1) % 32768;
   }
}
```

La boucle while

Une boucle while est une instruction de la forme :

while (e) p où e est une expression et p est une instruction, appelée *corps* de cette boucle.

Exécuter la boucle while (e) p a pour effet d'exécuter l'instruction p plusieurs fois tant que la valeur de l'expression e est égale à true.

SAVOIR-FAIRE Écrire un programme utilisant une boucle while

- Identifier la condition.
- Écrire le corps de la boucle.
- Prévoir une initialisation des variables en amont de la boucle et un post-traitement en aval.

La boucle while

Exercice 4 :

Rechercher une sous-chaîne dans une chaîne de caractères.
Par exemple, rechercher si une chaine de caractères donnée contient la sous-chaîne « oui ».

Correction exercice 4 :

```
void main() {
   String chaine;
   boolean trouve;
   chaine = readString("Saisir la chaine de caractères de départ");
   trouve = false;
   for (int i=0;i<chaine.length()-2 && !trouve;i++)
            if (chaine.charAt(i) == 'o' && chaine.charAt(i+1) == 'u' && chaine.charAt(i+2) == 'i')
            {
                     println("La chaine de caractères "' + chaine + "' contient 'oui'");
                     trouve = true;
            }
            if (!trouve)
            println("La chaine de caractères "' + chaine + "' ne contient pas 'oui'");
}
```

La boucle while

Exercice 5 : suite de Syracuse

On définit une suite *u* de la manière suivante :

- u_0 = 1000
- si u_n = 1, alors la suite est finie et u_n est son dernier élément
- si u_n est pair, alors $u_{n+1} = u_n / 2$
- si u_n est impair et distinct de 1, alors $u_{n+1} = 3\,u_n + 1$

Écrire un programme qui affiche les termes de la suite *u*.

Cette suite est-elle finie ?

Correction exercice 5 : suite de Syracuse

```
void main() {
   int u = 1000;
   int n = 0;
   while (u > 1) {
                println("u("+n+")="+u);
     if (u % 2 == 0) u = u / 2;
     else u = 3 * u + 1;
     n++;
   }
   println("u("+n+")="+u);
}
```

L'exécution du programme conduit à l'affichage de 112 termes jusqu'au dernier $u_{111} = 1$.

Remarque :
On ne sait pas si pour toute valeur de u_0 si la suite u est toujours convergente.

La boucle while

Exercice 6

Écrire un programme qui détermine le plus petit multiple commun à deux nombres entiers entrés au clavier.

La boucle while

Correction exercice 6

```
void main() {
   int a = readInt("Saisir le premier entier");
   int b = readInt("Saisir le deuxième entier");
   int x,y;
   int ppcm;
   boolean fini;
   if (a > b) {
      x = a;
      y = b;
   } else {
      x = b;
      y = a;
   }
   fini = false;
   ppcm = x;
   while ( ! fini) {
      if (ppcm % y == 0) fini = true;
      else ppcm = ppcm + x;
   }
   println("Le PPCM de " + a + " et de " + b + " est égal à " + ppcm);
}
```

La boucle while

La non-terminaison

Avec la boucle while apparaît un nouveau comportement possible pour les programmes : la non-terminaison.
Il est possible d'écrire une instruction while (e) p telle que la valeur de l'expression e soit toujours égale à true, si bien que l'exécution de l'instruction p se répète et se répète, sans que jamais l'exécution de la boucle ne se termine. Un exemple simple est le suivant :

```
while (true) { System.out.print("allô  ");}
```

qui affiche allô allô allô… sans jamais s'arrêter.

La boucle while

SAVOIR-FAIRE :
Choisir entre une boucle for et la boucle while pour écrire un programme.

Si on connaît à l'avance le nombre de répétitions à effectuer, la boucle for est toute indiquée. À l'inverse, si la décision d'arrêter la boucle ne peut s'exprimer que par un test, c'est la boucle while qu'il faut choisir.

Exercice 7

Quelle boucle est adaptée à l'écriture de programmes traitant les problèmes suivants :

- le calcul du total à payer à une caisse enregistreuse,
- la recherche du jour le plus pluvieux d'une année,
- le calcul du périmètre d'un polygone,
- le calcul de la durée d'une émission de radio, connaissant ses horaires de début et de fin ?

La boucle while

Correction exercice 7

Quelle boucle est adaptée à l'écriture de programmes traitant les problèmes suivants :

- le calcul du total à payer à une caisse enregistreuse,

Une boucle while *: on ne sait pas combien il y aura d'articles, on ne s'arrête que lorsque le tapis est vide.*

- la recherche du jour le plus pluvieux d'une année,

Une boucle for *: le corps de la boucle doit être répété 365 fois exactement*

- le calcul du périmètre d'un polygone,

Cela dépend : si le nombre de côtés est connu, une boucle for*, sinon, une boucle* while *qui s'arrête lorsqu'on est revenu au sommet de départ.*

- le calcul de la durée d'une émission de radio, connaissant ses horaires de début et de fin ?

Il n'y a pas besoin de boucle

La boucle while

Exercice 8

Dans cet exercice on écrit plusieurs versions d'un programme qui joue à la bataille navale, autrement dit qui cherche à couler un bateau. Comme dans le premier exemple, on suppose qu'il n'y a qu'un seul bateau d'une seule case, dont la position est connue par l'utilisateur.

a) Programmer l'algorithme naïf qui consiste à essayer toutes les cases systématiquement.

b) Améliorer cet algorithme pour qu'il s'arrête quand il a coulé le bateau.

c) Améliorer cet algorithme pour qu'il cherche le bateau intelligemment si celui-ci est en vue, c'est-à-dire dans une case adjacente au dernier essai effectué.

d) Peut-on encore améliorer cet algorithme pour minimiser le nombre d'essais nécessaires ?

La boucle while

Correction exercice 8

a)

```
void main() {
   int taille_grille;
   taille_grille = readInt("Taille de la grille : ");
   int pos_x_bateau = (int) (Math.random() * taille_grille);
   int pos_y_bateau = (int) (Math.random() * taille_grille);
   int x,y;
   int diff_x,diff_y;
   for (int i = 0; i < taille_grille; i ++) {
        for (int j=0;j <taille_grille;j++)
        {
             diff_x = Math.abs(i - pos_x_bateau);
             diff_y = Math.abs(j - pos_y_bateau);
             print("x = " + i + " y = " + j);
             if (diff_x == 0 && diff_y == 0) println(" Coulé !");
             else if (diff_x <= 1 && diff_y <= 1) println(" En vue");
             else println(" A l'eau !");
        }
   }
}
```

Correction exercice 8

a) Avec visualisation graphique de la grille

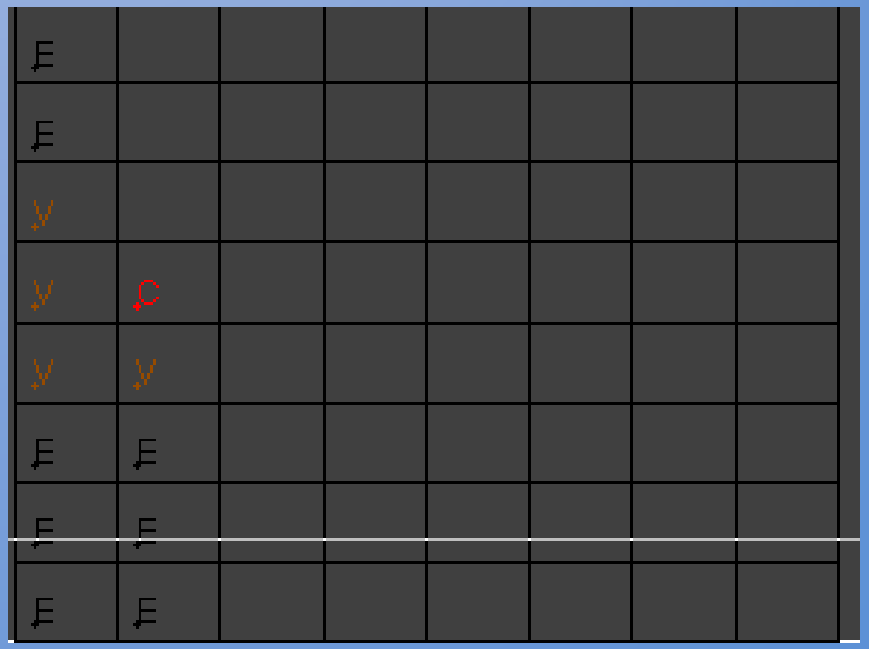
```
void main() {
   int taille_grille = readInt("Taille de la grille : ");
   int pos_x_bateau = (int) (Math.random() * taille_grille);
   int pos_y_bateau = (int) (Math.random() * taille_grille);
   int x,y,diff_x,diff_y;
   reset(taille_grille + 1, taille_grille + 1);
   for (int i = 0; i < taille_grille; i ++) {
      addLine(i, 0, i, taille_grille, 0);
      addLine(0, i, taille_grille, i, 0);
      for (int j = 0; j < taille_grille; j ++) {
         diff_x = Math.abs(i - pos_x_bateau);
         diff_y = Math.abs(j - pos_y_bateau);
         print("x = " + i + " y = " + j);
         if (diff_x == 0 && diff_y == 0) {
            println(" Coulé !");
            addString(i + 0.2, j + 0.2, "C", 2);
         } else if (diff_x <= 1 && diff_y <= 1) {
            println(" En vue");
            addString(i + 0.2, j + 0.2, "V", 1);
         } else {
            println(" A l'eau !");
            addString(i + 0.2, j + 0.2, "E", 0);
         }
      }
   }
   addLine(taille_grille, 0, taille_grille, taille_grille, 0);
   addLine(0, taille_grille, taille_grille, taille_grille, 0);
}
```

La boucle while

Correction exercice 8

b) Avec visualisation graphique de la grille

```
void main() {
  int taille_grille = readInt("Taille de la grille : ");
  int pos_x_bateau = (int) (Math.random() * taille_grille);
  int pos_y_bateau = (int) (Math.random() * taille_grille);
  int x,y,diff_x,diff_y;
  reset(taille_grille + 1, taille_grille + 1);
  boolean coule = false;
  for (int i = 0; i < taille_grille; i ++) {
    addLine(i, 0, i, taille_grille, 0);
    addLine(0, i, taille_grille, i, 0);
  }
  for (int i = 0; i < taille_grille && coule == false; i ++) {
    for (int j = 0; j < taille_grille && coule == false; j ++) {
      diff_x = Math.abs(i - pos_x_bateau);
      diff_y = Math.abs(j - pos_y_bateau);
      print("x = " + i + " y = " + j);
      if (diff_x == 0 && diff_y == 0) {
        println(" Coulé !");
        addString(i + 0.2, j + 0.2, "C", 2);
        coule = true;
      } else if (diff_x <= 1 && diff_y <= 1) {
        println(" En vue");
        addString(i + 0.2, j + 0.2, "V", 1);
      } else {
        println(" A l'eau !");
        addString(i + 0.2, j + 0.2, "E", 0);
      }
    }
  }
  addLine(taille_grille, 0, taille_grille, taille_grille, 0);
  addLine(0, taille_grille, taille_grille, taille_grille, 0);
}
```



La boucle while

Correction exercice 8

c) Avec visualisation graphique de la grille

```
void main() {
   int taille_grille = readInt("Taille de la grille : ");
   int pos_x_bateau = (int) (Math.random() * taille_grille);
   int pos_y_bateau = (int) (Math.random() * taille_grille);
   int x,y,diff_x,diff_y;
   int en_vue_x,en_vue_y;
   reset(taille_grille + 1, taille_grille + 1);
   boolean coule = false;
   boolean en_vue = false;
   for (int i = 0; i < taille_grille; i ++) {
      addLine(i, 0, i, taille_grille, 0);
      addLine(0, i, taille_grille, i, 0);
   }
   en_vue_x = - 1;
   en_vue_y = - 1;
```

La boucle while

Correction exercice 8

c) Avec visualisation graphique de la grille

```
for (int i = 0; i < taille_grille && ! coule && ! en_vue; i ++) {
      for (int j = 0; j < taille_grille && ! coule && ! en_vue; j ++) {
         diff_x = Math.abs(i - pos_x_bateau);
         diff_y = Math.abs(j - pos_y_bateau);
         print("x = " + i + " y = " + j);
         if (diff_x == 0 && diff_y == 0) {
            println(" Coulé !");
            addString(i + 0.2, j + 0.2, "C", 2);
            coule = true;
         } else if (diff_x <= 1 && diff_y <= 1) {
            println(" En vue");
            addString(i + 0.2, j + 0.2, "V", 1);
            en_vue = true;
            en_vue_x = i;
            en_vue_y = j;
         } else {
            println(" A l'eau !");
            addString(i + 0.2, j + 0.2, "E", 0);
         }
      }
   }
   if (en_vue) {
      for (int k = 0; k <= 1 && ! coule; k ++) for (int l = 0; l <= 1 && ! coule; l ++) {
         if ((k != 0 || l != 0) && (k + en_vue_x >= 0) && (k + en_vue_x) <= taille_grille && (l + en_vue_y >= 0) && (l + en_vue_y)
<= taille_grille) {
            diff_x = Math.abs(k + en_vue_x - pos_x_bateau);
```

La boucle while

Correction exercice 8

c) Avec visualisation graphique de la grille

```
if (en_vue) {
      for (int k = 0; k <= 1 && ! coule; k ++) for (int l = 0; l <= 1 && ! coule; l ++) {
         if ((k != 0 || l != 0) && (k + en_vue_x >= 0) && (k + en_vue_x) <= taille_grille && (l + en_vue_y >= 0)
&& (l + en_vue_y) <= taille_grille) {
            diff_x = Math.abs(k + en_vue_x - pos_x_bateau);
            diff_y = Math.abs(l + en_vue_y - pos_y_bateau);
            print("x = " + (k + en_vue_x) + " y = " + (l + en_vue_y));
            if (diff_x == 0 && diff_y == 0) {
               println(" Coulé !");
               addString(k + en_vue_x + 0.2, l + en_vue_y + 0.2, "C", 2);
               coule = true;
            } else if (diff_x <= 1 && diff_y <= 1) {
               println(" En vue");
               addString(k + en_vue_x + 0.2, l + en_vue_y + 0.2, "V", 1);
            } else {
               println(" A l'eau ! : erreur algorithme");
               addString(k + en_vue_x + 0.2, l + en_vue_y + 0.2, "E", 0);
            }
         }
      }
   }
   addLine(taille_grille, 0, taille_grille, taille_grille, 0);
   addLine(0, taille_grille, taille_grille, taille_grille, 0);
}
```