Introduction à la Programmation 1

Séance 8 de cours/TD

Université Paris-Diderot

Objectifs:

- Exécuter à la main des programmes contenant des boucles.
- Exécuter à la main des programmes contenant des fonctions.

1 Représentation mémoire des tableaux

Mémoire et tableau [Cours]

- Dans la mémoire, on stocke également les tableaux et leur contenu.
- Si a est une variable de type tableau référant un tableau créé, alors dans la mémoire, la variable a sera associée à la valeur $\$i$ où i est un entier positif. Le tableau correspondant sera représenté à côté de la mémoire des variables, dans une mémoire auxiliaire appelée le tas et qui est préservé par les appels de fonction et de procédure. Cette seconde mémoire associe donc des tableaux à des valeurs de la forme $\$i$.
- Par exemple :

$t1$	$t2$
$1	$2

$\$1 = \{2, 3, 4, 5\}$ $\$2 = \{-9, -3, 0\}$

  indique une mémoire où la variable t1 réfère le tableau $\{2,\ 3,\ 4,\ 5\}$ et la variable t2 le tableau $\{-9,\ -3,\ 0\}$
- Nous noterons $\square$ pour le contenu des cases non initialisées d'un tableau. Cela signifie que la case contient bien une valeur mais que cette valeur n'a pas été fixée par le programme et ne peut donc pas être exploitée de façon sûre.
- Ainsi après l'instruction « `int [] t = new int[4];` », la mémoire sera :

t
$1

$\$1 = \{\square, \square, \square, \square\}$

- Si une expression accède aux valeurs du tableau (par exemple dans « `a[2]` ») alors pour il faut aller regarder dans la mémoire quel est le tableau référé par la variable a, par exemple $\$3$ et ensuite prendre la valeur qui se trouve dans la troisième case du tableau $\$3$.
- Si un tableau $\$i$ n'est référencé par aucune variable dans la mémoire, alors on peut le supprimer.

Exercice 1 (Comprendre la représentation des tableaux, $\star$)
Étant donnée la mémoire suivante ;

$t1$	$t2$	i	j
$1	$2	1	3

$\$1 = \{0, 2, 4, 6, 8, 10\}$ $\$2 = \{100, 101, 110, 111\}$

quelle est la valeur des expressions « `t1[1]` », « `t2[i]` », « `t1[i] - t2[j] + i` », « `t1[i + j]` » ? $\square$

Exercice 2 (Exécution d'un programme avec des tableaux, $\star$)

Décrire sur papier l'évolution de la mémoire lors de l'exécution du programme suivant :

```java
public class ExoExecTab {
    public static void main (String[] args) {
        int[] t1 = { -20, 3, 5, 9, 12};
        int[] t2 = new int[2];
        t2[0] = 1;
        if (t1[2] > 6) {
            t2[1] = t1[2];
        } else {
            t2[1] = t1[1];
        }
    }
}
```

code/ExoExecTab.java

□

Représentation des tableaux de tableaux [Cours]

- Avec la représentation précédente, il est également possible de représenter des tableaux de tableaux.
- Dans les cases d'un tableau de tableaux, on trouvera des éléments précédés du symbole $ faisant référence à d'autres tableaux.
- Par exemple, dans la mémoire suivante :

t
$1

$\$1 = \{\$2, \$3, \$4\}$ $\$2 = \{1, 2\}$ $\$3 = \{10, 20, 30\}$ $\$4 = \{100, 200, 300, 400\}$

la variable t réfère un tableau de tableaux contenant dans l'ordre les trois tableaux suivants : {1, 2}, {10, 20, 30} et {100, 200, 300, 400}

Exercice 3 (Création de tableaux, ⋆⋆)
Donner un programme permettant d'aboutir à la mémoire suivante :

t
$1

$\$1 = \{\$2, \$3, \$4\}$ $\$2 = \{1, 2\}$ $\$3 = \{10, 20, 30\}$ $\$4 = \{100, 200, 300, 400\}$

□

2 Exécution des boucles While

Boucles avec while [Cours]

- Pour les boucles while, on commence par évaluer le test et on décide ensuite selon la valeur (vraie ou fausse) du test, la ligne suivante.
- Lorsque l'on arrive à la fin du corps de la boucle, on revient à la ligne où le test est effectué.

Exemple d'exécutions [Cours]

Pour le programme ci-dessous :

```java
public class ExoBoucleWhile {
    public static void main (String[] args) {
        int x = 4;
        int y = 1;
        int z = 7;
        while (x > 0) {
            y = 5 * y;
```

```
8                 x = x - 2;
9             }
10            z = y - 1;
11        }
12    }
```

code/ExoBoucleWhile.java

On a la séquence d'exécution suivante :

PC	x		
3	4		
PC	x	y	
4	4	1	
PC	x	y	z
5	4	1	7
PC	x	y	z
6	4	1	7
PC	x	y	z
7	4	5	7
PC	x	y	z
8	2	5	7
PC	x	y	z
9	2	5	7
PC	x	y	z
6	2	5	7
PC	x	y	z
7	2	25	7
PC	x	y	z
8	0	25	7
PC	x	y	z
9	0	25	7
PC	x	y	z
6	0	25	7
PC	x	y	z
10	0	25	24

Précis d'exécution : while [Cours]

- Étant donnée une configuration avec mémoire M et compteur de programme i.
- Si on trouve à l'adresse i la tête d'une boucle "while (e) {" :
  - On commence par évaluer la condition e par rapport à mémoire M.
  - Si le résultat est true alors le nouveau compteur de programme est $i + 1$.
  - Si le résultat est false alors le nouveau compteur de programme est la ligne qui suit la fin de la boucle.
- Si on trouve à l'adresse i une ligne l'accolade "}" qui met un terme à l'itération courante ("ferme la boucle while"), alors le nouveau compteur de programme est le numéro de ligne qui contient la tête de la boucle correspondante.
- Aucune de ces lignes ne modifie la mémoire.

Exercice 4 (Exécution d'un programme avec while, ⋆)

Décrire sur le papier l'exécution du programme suivant :

```
1  public class ExoBoucleWhile2 {
2      public static void main (String[] args) {
3          int[]tab = {0, 1, 2, 3};
4          int i = 0;
5          boolean flag = false;
```

```
6           while (i < 3 && flag == false) {
7               if (tab[i] == 1) {
8                   flag = true;
9               }
10              i = i + 1;
11          }
12      }
13  }
```

code/ExoBoucleWhile2.java

□

3 Exécution des boucles For

Boucles avec for [Cours]

- Le principe est similaire à celui des boucles while.
- Il ne faut pas oublier de faire la mise à jour de la variable utilisée dans la boucle lorsque l'on a atteint le bas de la boucle utilisée dans le for.
- Il faut penser à enlever la variable d'itération de la mémoire.
  (Il s'agit de la variable déclarée dans la partie initialisation de la boucle.)

Précis d'exécution for [Cours]

- Étant donnée une configuration avec mémoire M et compteur de programme i.
- Si on trouve à l'adresse i la tête d'une boucle « for (init ;cond;incr) { » :
  - Si on est arrivé à la ligne i en venant du "haut", on exécute $init$, ce qui donne la nouvelle mémoire.
  - Si on est arrivé à la ligne i en venant du "bas", on exécute $incr$, ce qui donne la nouvelle mémoire.
- Si l'évaluation de $cond$ par rapport à la nouvelle mémoire donne true alors le nouveau compteur de programme est $i + 1$. Si l'évaluation de $cond$ donne false alors le nouveau compteur de programme est la première ligne suivant la boucle, et on supprime de la mémoire la variable déclarée pour la boucle.
- Si on trouve à l'adresse i une ligne l'accolade "}" qui met un terme à l'itération courante ("ferme la boucle for"), alors le nouveau compteur de programme est le numéro de ligne qui contient la tête de la boucle correspondante.

Exemple d'exécutions [Cours]

Pour le programme ci-dessous :

```
1  public class ExoBoucleFor {
2      public static void main (String[] args) {
3          int s = 1;
4          for (int i = 0; i < 3; i++) {
5              s = s * 2;
6          }
7      }
8  }
```

code/ExoBoucleFor.java

On a la séquence d'exécution suivante :

PC	s
3	1

PC	s	i
4	1	0

PC	s	i
5	2	0
PC	s	i
6	2	0
PC	s	i
4	2	1
PC	s	i
5	4	1
PC	s	i
6	4	1
PC	s	i
4	4	2
PC	s	i
5	8	2
PC	s	i
6	8	2
PC	s	i
4	8	3
PC	s	
7	8	

Exercice 5 (Exécution d'un programme avec boucle for, ⋆)
Décrire sur le papier l'exécution du programme suivant :

```java
public class ExoBoucleFor2 {
    public static void main (String[] args) {
        String st = "";
        for (int j = 1; j < 5; j++) {
            st = st + " ab";
        }
    }
}
```

code/ExoBoucleFor2.java

□

Exercice 6 (Exécution d'un programme avec boucle for et tableaux, ⋆)
Décrire sur le papier l'exécution du programme suivant :

```java
public class ExoBoucleForTab {
    public static void main (String[] args) {
        int []t = {2, 4, 6, 8};
        int sum = 0;
        for (int j = 0; j < 4; j++) {
            sum = sum + t[j];
        }
        sum = sum / 2;
    }
}
```

code/ExoBoucleForTab.java

□

4 Exécution des fonctions

Appels de fonctions [Cours]

- Dans ce cadre, nous supposerons que tous les retours des fonctions sont stockés dans une variable. (Ainsi, tous les appels de fonction seront de la forme y=f(...).)
- Lors de l'appel à une fonction :
  - Il faut savoir où reprendre lorsque la fonction finira son exécution (quand elle fera return par exemple).

    ⇒ Pour cela on garde le compteur de programmes où se fait l'appel de fonction en mémoire.
  - Il faut un compteur de programme pour l'exécution de la fonction

    ⇒ Pour cela, on rajoute un compteur de programme à droite de l'ancien compteur en les séparant par le symbole $\triangleright$.
  - On créé une nouvelle zone mémoire pour l'exécution de la fonction, cette nouvelle zone étant séparée de l'ancienne par $||$. Elle sert à stocker la valeur des paramètres de la fonction et des variables locales à cette fonction.

Exemple d'exécution [Cours]

Pour le programme ci-dessous :

```java
public class ExoFunc {

    public static int square (int a) {
        int d = a * a;
        return d;
    }

    public static void main (String[] args) {
        int x = 4;
        int y = 1;
        y = square (x);
        x = 2 * y - 1;
    }
}
```

code/ExoFunc.java

On a la séquence d'exécutions suivante :

PC	x
9	4

PC	x	y
10	4	1

PC	x	y	a
$11 \triangleright 3$	4	1	4

PC	x	y	a	d
$11 \triangleright 4$	4	1	4	16

PC	x	y
$11 \triangleright 5$	4	16

PC	x	y
12	31	16

Précis d'exécution : appels de fonctions [Cours]

- Étant donnée une configuration avec mémoire M et compteur de programme i.
- Si l'adresse i est un appel de fonction « x = f (...) » d'en-tête « T f (T1 x1, ..., TN xN) »
  - Pour chaque paramètre (x1 ,... xN), on crée à droite de la table en séparant par || les nouveaux paramètres et les valeurs données à ces paramètres au moment de l'appel.
  - Dans la table, au niveau du compteur de programme au rajoute $i \triangleright j$ où j est la ligne où se trouve l'entête de la fonction f.
- Ensuite, on exécute la fonction de la même façon que le programme en prenant comme compteur de programme, la valeur la plus à droite du symbole $\triangleright$
- Si l'on a dans la configuration, un compteur $i \triangleright k$ et que l'on voit « return a », alors :
  - On regarde à la ligne i du programme (où il y a l'appel de la fonction qu'on vient de terminer) quelle est la variable à laquelle le résultat de la fonction doit être affecté, et on lui affecte la valeur de l'expression a.
  - On supprime de la table les variables les plus à droite du séparateur ||.
  - Le compteur de programme devient $i + 1$.

Exercice 7 *Exécution d'un programme avec appel de fonctions Décrire sur le papier l'exécution du programme suivant :'*

```java
public class ExoFunc2 {

    public static int substraction (int a, int b) {
        int d = a - b;
        return d;
    }

    public static void main (String[] args) {
        int x = 10;
        int y = 5;
        int r = 0;
        if (y > x) {
            r = substraction (y, x);
        } else {
            r = substraction (x, y);
        }
    }
}
```

code/ExoFunc2.java

□

5 DIY

Exercice 8 (Affichage, ⋆⋆)

Après avoir exécuté sur papier le programme suivant jusqu'à la ligne 30, en déduire ce qu'il affichera.

```java
public class ExoMystery {

    // Affiche un entier suivi d'un retour à la ligne.
    public static void printIntLn (int i) {
        System.out.println (i);
    }

    public static int f (int x) {
        if (x * x == x) {
```

```
10              return 1;
11          }
12          return 2;
13      }
14
15      public static void main (String[] s) {
16          int x = 3;
17          int y = 4;
18          int z = 0;
19          int i = 0;
20          boolean a = (x != 0) && ! (x - y == -1 || x * y != 12);
21          y = y - 1;
22          if (a) {
23              i = 2;
24          } else {
25              i = y;
26          }
27          for (x = 0; x <= y; x++) {
28              z = f (x);
29          }
30          printIntLn (i);
31          printIntLn (x);
32          printIntLn (y);
33          printIntLn (z);
34      }
35  }
```

code/ExoMystery.java

□

Exercice 9 (Carré, ⋆)

Écrire une fonction qui prend en paramètre un tableau d'entiers T à deux dimensions et le modifie directement en élevant chacun de ses éléments au carré. □

Exercice 10 (Cumul, ⋆⋆)

On s'intéresse aux précipitations d'une année. Un tableau d'entiers P à deux dimensions contient la quantité de pluie tombée chaque jour : $P[m][j]$ est le nombre de millimètres d'eau tombée le jour j du mois m. Écrire une fonction `cumul` qui modifie directement le tableau P pour représenter la quantité de pluie cumulée depuis le début de chaque mois, c'est-à-dire que $P[m][j]$ est maintenant la quantité d'eau tombée entre le début du mois m et le jour j (inclus) du mois m.

Ainsi, si les valeurs des 5 premiers jours de février sont | 0 | 10 | 2 | 0 | 5 |*, alors on obtiendrait les nouvelles valeurs* | 0 | 10 | 12 | 12 | 17 |*.* □

Exercice 11 (Tableaux, ⋆⋆)

(Les trois questions suivantes sont indépendantes.)

1. *Écrire une fonction qui prend en entrée un tableau d'entiers `t` à 2 dimensions, et qui renvoie un tableau contenant deux éléments : le minimum et le maximum de `t`.*
2. *Écrire une fonction qui prend en entrée un tableau d'entiers `t` à 2 dimensions, et qui modifie `t` directement en remplaçant chaque entier par sa valeur absolue.*
3. *Écrire une fonction qui prend en entrée un tableau `t` à une dimension, contenant des entiers non nécessairement distincts, et qui renvoie l'élément apparaissant le plus souvent dans `t`. Par exemple, sur le tableau $\{5, 2, 3, 3, 2, 5, 1, 5, 1\}$, on doit renvoyer 5.*

□

Exercice 12 (Deviner, ⋆⋆)

Donner un programme avec une boucle while *dont la séquence d'instructions est la suivante.*

PC	x	
3	4	
PC	x	y
4	4	1
PC	x	y
5	4	1
PC	x	y
6	4	4
PC	x	y
7	3	4
PC	x	y
8	3	4
PC	x	y
5	3	4
PC	x	y
6	3	12
PC	x	y
7	2	12
PC	x	y
8	2	12
PC	x	y
5	2	12
PC	x	y
6	2	24
PC	x	y
7	1	24
PC	x	y
8	1	24
PC	x	y
9	1	24

□

Exercice 13 (Polynômes, ⋆⋆⋆)

On rappelle qu'un polynôme est une fonction p de la forme

$$p(x) = \alpha_d x^d + \alpha_{d-1} x^{d-1} + \cdots + \alpha_1 x + \alpha_0.$$

Ainsi, on peut représenter un polynôme p par ses coefficients $\alpha_0, \ldots, \alpha_d$, par exemple stockés dans un tableau `p` *de taille $d+1$ dont la i-ème case contient la valeur α_i. On suppose que les coefficients α_i sont entiers. Par exemple, le polynôme $p(x) = 5x^4 - 3x^2 + x - 1$ sera représenté par le tableau $\{-1, 1, -3, 0, 5\}$.*

On se propose d'écrire des fonctions de manipulation de polynômes grâce à cette représentation en tableau.

1. *Écrire une fonction* `ask_polynom` *qui demande à l'utilisateur un entier d puis $d+1$ coefficients $\alpha_0, \ldots, \alpha_d$, et renvoie le tableau* `p` *correspondant au polynôme $p(x) = \alpha_d x^d + \alpha_{d-1} x^{d-1} + \cdots + \alpha_1 x + \alpha_0$.*
2. *Écrire une fonction* `show_polynom` *qui prend en paramètre un tableau* `p` *représentant un polynôme p, et qui affiche à l'écran le polynôme p. Par exemple, sur l'entrée $[-3, 2, 0, 1]$ on affichera* `x^3+2x-3`*.*
3. *Écrire une fonction* `evaluate` *qui prend en entrée un tableau* `p` *représentant un polynôme p et un entier a, et qui renvoie la valeur $p(a)$ de p évalué en a.*
4. *Pour ajouter deux polynômes p et q, il suffit d'ajouter les coefficients correspondant aux termes de même degré. Écrire une fonction* `sum` *qui prend en entrée deux tableaux* `p` *et* `q` *représentant respectivement les polynômes p et q, et qui renvoie un tableau* `r` *correspondant au polynôme $r = p + q$.

5. *(Plus difficile)*
   Soient deux polynômes $p(x) = \alpha_d x^d + \alpha_{d-1} x^{d-1} + \cdots + \alpha_1 x + \alpha_0$ *et* $q(x) = \beta_e x^e + \beta_{e-1} x^{e-1} + \cdots + \beta_1 x + \beta_0$. *Soit* $r = pq$ *le produit de* p *et* q. *Alors on peut exprimer les coefficients* γ_i *de* r *comme suit :*

$$\gamma_i = \sum_{j+k=i} \alpha_j \beta_k.$$

   Par exemple, si $p(x) = 5x^2 + 3x + 1$ *et* $q(x) = x^3 - x$, *alors* $r(x) = 5x^5 + 3x^4 - 4x^3 - 3x^2 - x$.
   Écrire une fonction `product` *qui prend en entrée deux tableaux* `p` *et* `q` *représentant respectivement les polynômes* p *et* q, *et qui renvoie un tableau* `r` *correspondant au polynôme* $r = pq$.

6. *En utilisant les fonctions précédentes, écrire un programme qui demande à l'utilisateur d'entrer les coefficients de deux polynômes* p *et* q, *ainsi qu'un entier* a, *puis qui affiche* p *et* q, *les valeurs* $p(a)$ *et* $q(a)$, *la somme* $p + q$ *et enfin le produit* pq.

□

Exercice 14 (Modification de tableau, ⋆)
Écrire une fonction qui prend en paramètre un tableau d'entiers T *à deux dimensions et le modifie directement comme suit : elle ajoute* 1 *à chaque élément négatif et retranche* 1 *à chaque élément positif.* □