Département d'Informatique
DUT Informatique S1
Année 2014/2015
Algorithmique et programmation

Algorithmique et programmation

Exercices de TD

Table des matières

Version du 7 décembre 2014.

1 Séquences d'actions

1.1 Écluse

Écrire l'*idée de l'algorithme* permettant de modéliser le franchissement d'une écluse par un bateau.

Les portes d'une écluse sont munies de vannes que l'on peut ouvrir ou fermer et qui permettent de laisser passer l'eau afin de mettre l'écluse à niveau (figure 1). Les portes sont également équipées de feux pouvant être vert ou rouge, pour autoriser ou non le passage du bateau.

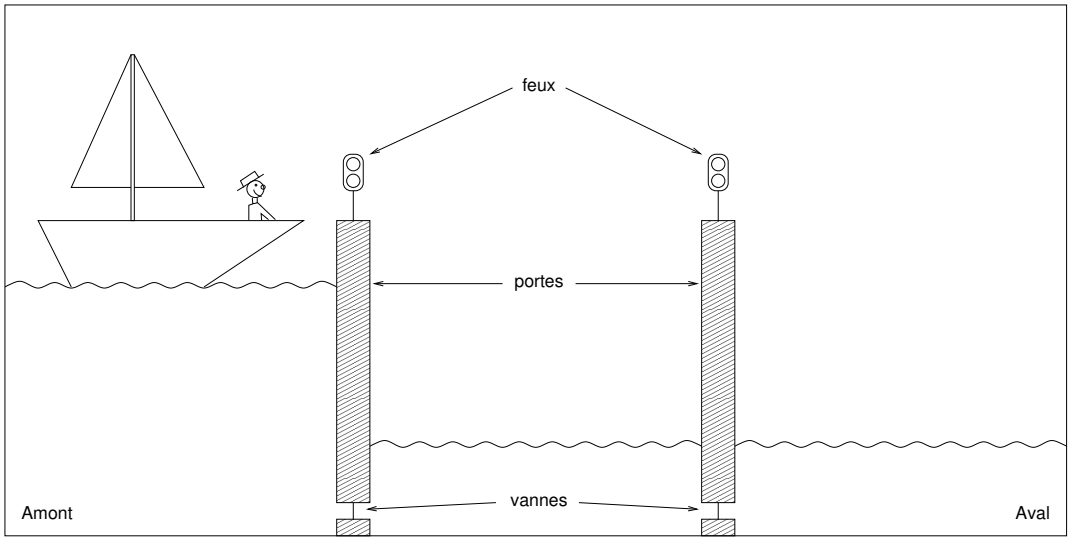

FIGURE 1 – Schéma d'un écluse

On considérera dans un premier temps que le bateau descend la rivière et que lorsqu'il se présente devant l'écluse, celle-ci est en niveau bas.

1.2 Radiateurs

1. Concevoir un algorithme pour calculer le nombre de radiateurs dont on a besoin pour chauffer une pièce. On sait qu'un radiateur est capable de chauffer $8\,\mathrm{m}^3$. L'utilisateur donnera la longueur, la largeur et la hauteur de la pièce en mètres.
2. Détailler le déroulement de l'algorithme pour une pièce de dimensions $6 \times 4 \times 2{,}5$ m.
3. Traduire l'algorithme dans un programme Java complet.

1.3 Gazon

On veut calculer la surface du gazon et le temps de tonte du gazon pour un terrain rectangulaire sur lequel sont bâtis une maison rectangulaire et un appentis triangulaire, et qui comporte une piscine circulaire. On sait que la tonte du gazon prend x secondes au m^2 (x est donné), et on donne les dimensions indiquées sur la figure 2.

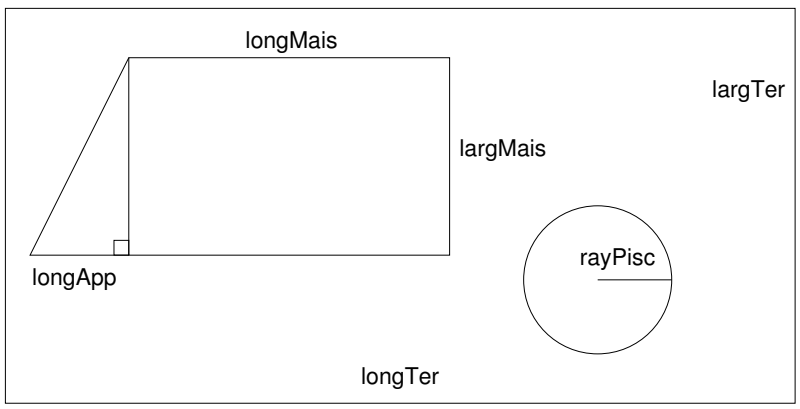

FIGURE 2 – Schéma du terrain à tondre

1.4 Heures, minutes, secondes

Écrire un algorithme pour convertir un nombre de secondes en un nombre d'heures, de minutes et de secondes. On utilisera les opérateurs modulo et division entière.

1.5 Programme Gazon

Utiliser les algorithmes précédents pour écrire un programme Java qui calcule et affiche surface de gazon et temps de tonte. Les informations nécessaires seront demandées à l'utilisateur. Le temps de tonte sera affiché en heures, minutes, secondes.

Indications

En Java,

- l'opération modulo (mod) se note `%`. Par exemple : `duree % 3600`.
- la constante mathématique π est définie par `Math.PI`.

2 Conditionnelles

2.1 Dérouler un algorithme

Réalisez l'exécution de l'algorithme suivant en donnant successivement les valeurs 20, 150, 50, 51 et 400 à *valeur*.

Lexique des variables

valeur (entier) entier donné DONNÉE

Algorithme

```
si valeur > 100 alors
|  écrire "A"
|  si valeur > 200 alors
|  |  écrire "B"
|  sinon
|  |  écrire "C"
|  fsi
sinon
|  si valeur > 50 alors
|  |  écrire "D"
|  sinon
|  |  écrire "E"
|  fsi
fsi
```

2.2 Pizzas

On ne fait pas forcément des économies en achetant plus gros. Est-ce vrai quand on achète des pizzas ? Concevoir un algorithme qui lit le diamètre de deux pizzas, leur numéro et leur prix puis qui imprime le numéro de celle qui a le meilleur rapport *taille/prix*.

2.3 Coordonnées

Écrire un algorithme qui, étant donné les coordonnées x et y d'un point, détermine dans quelle partie (A, B, C ou D) du plan se trouve le point (cf. figure 3).

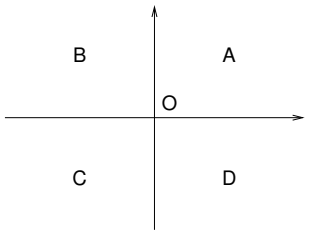

FIGURE 3 – Parties du plan

Pour aller plus loin. Écrire un algorithme de conversion des coordonnées du point en coordonnées polaires. La valeur de l'angle doit être dans l'intervalle $[0 ; 2\pi[$.

2.4 Mois de l'année

Écrire un algorithme qui demande un numéro de mois à l'utilisateur et indique en retour son nom et le nombre de jours dans ce mois.

3 Itérations

3.1 Lignes d'étoiles

1. Concevoir un algorithme qui, pour un caractère imprimable et un nombre n donnés, imprime une barre horizontale de n de ces caractères.

   ```
   ****************
   ```

2. Modifier l'algorithme pour l'impression d'une barre double.

   ```
   ****************
   ****************
   ```

3. Modifier l'algorithme pour l'impression d'une barre d'épaisseur quelconque donnée.
4. (optionnel) Transformer les algorithmes ci-dessus en fonctions.
5. Écrire un programme Java implémentant la dernière version de l'algorithme (épaisseur quelconque).

3.2 Carrés et cubes

Écrire un algorithme qui imprime la suite des carrés et des cubes des entiers compris entre 10 et 100.

3.3 Triangle de nombres

Concevoir un algorithme qui imprime pour n donné :

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
...........
............
.............
1 2 3 4 5 6 ... n
```

3.4 Tables de multiplications

Concevoir un algorithme qui imprime l'ensemble des tables de multiplication par 1, 2, ..., 12.
Modifier ensuite l'algorithme pour qu'il imprime 3 tables côte à côte.

3.5 Puissances

Imprimer la suite des puissances de x (nombre réel donné) : x, x^2, ..., x^n, avec n un entier positif donné. La solution ne devra pas utiliser d'opération « puissance ».

3.6 Suite alternée

Imprimer les termes de la suite alternée :

$$1, -2!, 3!, -4!, \ldots, (-1)^{n+1} \times n!$$

3.7 Sommes de fractions

1. Écrire un algorithme qui, pour n donné, calcule la somme :

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

2. Même question avec :

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^n}$$

3.8 Hauteurs de pluie

Écrire un algorithme qui lit les hauteurs de pluie (en mm) tombée durant un an (de janvier à décembre), puis imprime la hauteur de pluie la plus forte ainsi que le numéro du mois où cela s'est produit, la hauteur de pluie la plus faible ainsi que le numéro du mois où cela s'est produit et la moyenne des hauteurs de pluie tombées par mois. On suppose que les hauteurs de pluie tombée sont différentes chaque mois et l'on ne vérifie pas ce fait.

3.9 Puissance de 2

On donne un réel x supposé positif, trouver le plus petit entier n tel que 2^n soit plus grand que x.

3.10 Chute libre

On veut imprimer les effets de la gravité sur un objet qui tombe depuis une tour de hauteur donnée en mètres. Pour cela, on imprime la hauteur à laquelle l'objet se trouve toutes les x (réel) secondes pendant sa chute. On sait que la distance d parcourue par un objet en fonction du temps de parcours t est donnée par la formule :

$$d = \frac{1}{2}gt^2$$

où g est la constante de gravitation (9,806 65). On écrira le message « boum » quand l'objet heurte le sol.

4 Fonctions

4.1 Calcul d'âge

Écrire une fonction qui calcule le nombre d'années (entières) qui s'écoulent entre deux dates (jour, mois, année).

L'utiliser dans un algorithme qui doit imprimer l'âge d'une personne étant donné sa date de naissance et indique sous forme de message si la personne est mineure, adulte ou peut bénéficier de la carte Vermeil (plus de 60 ans) !

4.2 Multiplication récursive

Écrire une fonction récursive qui fait le produit de deux nombres entiers positifs en utilisant des additions. L'idée est que :

$$a \times b = \begin{cases} 0 & \text{si } a = 0 \\ (a-1) \times b + b & \text{si } a > 0 \end{cases}$$

Exécuter la fonction pour $a = 4$ et $b = 5$ (pour cela, présenter les appels récursifs successifs sous forme d'un *arbre*), puis pour $a = 5$ et $b = 4$. Pour quel appel l'arbre est-il le plus haut ? Noter sur quel argument porte la récursion. Conclure.

4.3 Calculs de puissances

1. Écrire une fonction récursive qui calcule x^n sachant que :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ x^{n-1} \times x & \text{si } n > 0 \end{cases}$$

2. Écrire une fonction récursive qui calcule x^n sachant que :

$$x^n = \begin{cases} 1 & \text{si } n = 0 \\ \left(x^{n/2}\right)^2 & \text{pour } n > 0 \text{ pair} \\ \left(x^{(n-1)/2}\right)^2 \times x & \text{pour } n > 0 \text{ impair} \end{cases}$$

3. Comparer les exécutions (avec *arbre* des appels) des deux fonctions ci-dessus pour $n = 11$ et $x = 2$. Quel est l'arbre le plus court ? Qu'en conclure ?

4.4 Tours de Hanoï

Autrefois, à Hanoï, l'empereur a conçu un puzzle de n disques et trois piquets : A (départ), B (échangeur) et C (arrivée). Les disques étaient tous de tailles différentes et avaient un trou au milieu afin que l'on puisse passer les disques dans les piquets. À cause de leur poids important, on ne pouvait pas mettre un disque plus lourd au dessus de disques qui étaient plus petits. Le but du jeu est de transférer tous les disques du piquet A au piquet C, en ne déplaçant qu'un seul disque à la fois. Concevoir une solution récursive qui imprime un message décrivant la solution.

4.5 Flocon de Von Koch

Une famille de fonctions mathématiques donne des représentations graphiques intéressantes, il s'agit des fractales.

Le principe de dessin d'une famille de fractales est de remplacer chaque segment de la figure initiale par un motif et de réitérer le processus sur les segments de la figure obtenue. On arrête le processus en fixant le nombre maximal de subdivisions à effectuer (niveau de récursion).

Dans l'exemple de la fractale de Von Koch (cf. figure 4), on considère l'ajout des points C, D et E dans la figure initiale $[AB]$, en considérant que les longueurs des nouveaux segments $[AC]$, $[CD]$, $[DE]$ et $[EB]$ sont toutes égales à $1/3$ de la longueur du segment initial $[AB]$.

1. Écrire un algorithme permettant de calculer les coordonnées C, D et E, à partir des coordonnées des points A et B.

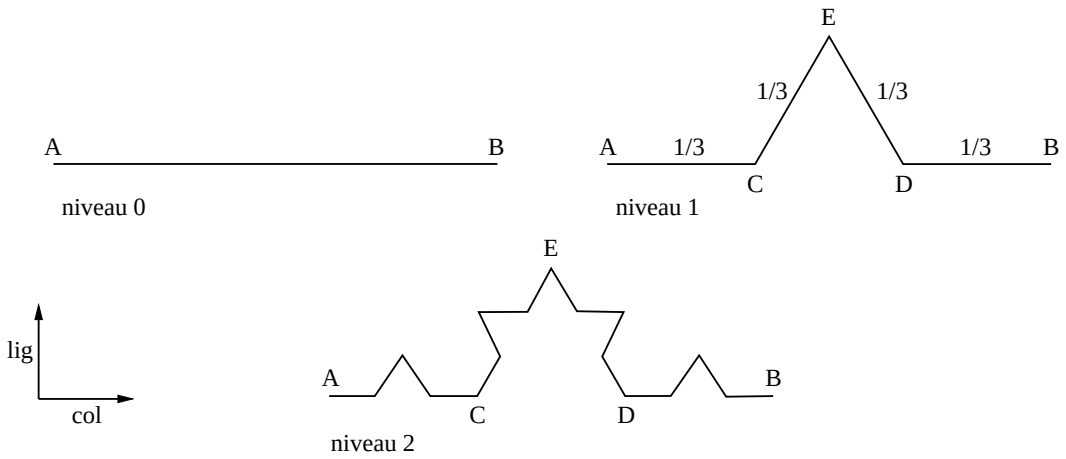

FIGURE 4 – Exemple de la fractale de Von Koch

2. Écrire une fonction récursive de dessin de la fractale de Von Koch pour deux points de départ et un niveau de récursion donnés.

   Pour effectuer le dessin, on considère que l'on dispose d'une fonction qui trace une ligne entre deux points donnés :

---

fonction ligne(**in** x_1 : entier, **in** y_1 : entier, **in** x_2 : entier, **in** y_2 : entier) : **vide**

Trace une ligne entre les points (x_1, y_1) et (x_2, y_2).

---

5 Tableaux

5.1 Notes

Écrire un algorithme qui, étant donné les notes des 30 élèves d'une classe, calcule le nombre de ceux qui ont obtenu une note supérieure à la moyenne et le nombre de ceux qui ont obtenu une note inférieure à la moyenne.

Question subsidiaire Résoudre le même problème, mais en utilisant la moyenne de la classe.

5.2 Températures

Écrire un algorithme qui, étant donné un relevé de températures sur un mois de 30 jours, imprime le ou les jours du mois où il y a eu le plus grand écart par rapport à la moyenne de ces températures.

5.3 Vote

Une organisation vote pour renouveler son directeur. Il y a 5 candidats numérotés de 1 à 5. Les noms des candidats sont, dans l'ordre de leur numéros : Ariane, Jean, Lucie, Marc, et Michel. Chaque membre de l'organisation envoie un bulletin de vote qui porte le numéro qui correspond au candidat de son choix. Écrire un algorithme pouvant servir à faire le décompte des voix, et afficher le ou les noms des candidats ayant le plus de voix.

5.4 Mélange

Écrire une fonction permettant de mélanger les valeurs d'un tableau d'entiers. On suppose que l'on dispose d'une fonction de tirage aléatoire :

---

fonction nombreAléatoire(**in** a : entier, **in** b : entier) : **ret** entier
Retourne un nombre entier tiré aléatoirement dans l'intervalle $[a\ ;\ b[$.

5.5 Nombres en toutes lettres

Écrire un algorithme qui lit des nombres entiers supposés positifs et inférieurs à 99999 et les imprime en toutes lettres. Par exemple, avec 367, on imprime *trois cent soixante-sept*. On utilisera des tableaux de constantes chaînes de caractères appropriés.

5.6 Recherche dichotomique

Exécuter la trace de l'algorithme de recherche dichotomique appliqué aux recherches successives des entiers 6, 1, 12, 30, 0, 21, 29 dans un tableau d'entiers contenant les valeurs 1, 5, 9, 12, 15, 21, 29. Vous vous appuierez sur les deux versions (récursive et itérative) données en cours.

5.7 Reversi

L'algorithme doit simuler un jeu appelé *Reversi*. Le jeu commence par afficher les chiffres de 1 à 9 dans le désordre. Le joueur donne un entier compris entre 1 et 9 qui indique le nombres de chiffres à inverser à partir de la gauche. La nouvelle configuration du jeu est affichée. Par exemple pour la configuration :

5 4 6 2 1 7 9 8 3

si le joueur donne l'entier 5, la nouvelle configuration sera :

1 2 6 4 5 7 9 8 3

Le jeu s'arrête quand on atteint une configuration où les chiffres sont placés par ordre croissant. L'objectif du joueur est de réaliser ce but avec le moins de tours possible. L'algorithme permettra d'imprimer en combien de tours le joueurs a réalisé le tri et lui permettra de recommencer le jeu sur la même configuration initiale autant de fois qu'il le désire. La configuration initiale est supposée fournie par une fonction dont on ne décrira pas l'algorithme :

---

fonction initTab(**out** *tab* : tableau d'entiers, **in** *taille* : entier) : **ret** vide
Remplit le tableau *tab* avec les nombre de 1 à *taille*, dans le désordre.

---

6 Tableaux 2D

6.1 Transposée de matrice

Écrire un algorithme calculant la transposée d'une matrice $m \times n$ de réels.

6.2 Matrice symétrique

Écrire un algorithme permettant de vérifier si une matrice carrée $n \times n$ est symétrique.

6.3 Somme et produit de matrices

Écrire un algorithme permettant de calculer la somme de deux matrices de réels. Écrire ensuite un algorithme permettant de calculer le produit de deux matrices de réels.

Rappels :

- La somme de deux matrices A et B de même taille $m \times n$ donne une matrice C de taille $m \times n$ également, calculée par :
$$C_{i,j} = A_{i,j} + B_{i,j} \qquad 1 \leq i \leq m, 1 \leq j \leq n$$
- Le produit de deux matrices A de taille $m \times n$ et B de taille $n \times p$ donne une matrice C de taille $m \times p$, calculée par :
$$C_{i,j} = \sum_{k=1}^{n} A_{i,k} \times B_{k,j} \qquad 1 \leq i \leq m, 1 \leq j \leq p$$

6.4 Jeu de la vie

Le *jeu de la vie* est un automate cellulaire imaginé par John Horton Conway en 1970. Malgré des règles très simples, le jeu de la vie permet le développement de motifs extrêmement complexes.

Dans ce jeu, on dispose d'un certain nombre de cellules, organisées en grille torique. Chaque cellule a donc 8 voisins. Les cellules du bords ont pour voisines celle du bord opposé. Une cellule a deux états possibles : morte ou vivante.

À chaque itération, l'état de toutes les cellules est mis à jour selon les règles suivantes :

naissance : une cellule morte qui a exactement 3 voisines vivantes devient vivante.

isolement : une cellule vivante qui a moins de 2 voisines vivantes meurt.

étouffement : une cellule vivante qui a plus de 3 voisines vivantes meurt.

L'état des autres cellules ne change pas.

Écrire un algorithme permettant de simuler l'évolution des cellules. On se contentera ici de calculer l'état des cellules, et on ne s'occupera pas d'un affichage éventuel.

Remarque : Au départ, l'état des cellules sera tiré aléatoirement (vivante ou morte, avec un probabilité de 1/2).

Question subsidiaire Modifier l'algorithme pour ne pas recalculer entièrement à chaque itération le nombre de voisines vivantes pour chaque cellules. Lorsqu'une cellule change d'état, le nombre de voisines de ses voisines est modifié.