Université Grenoble Alpes
U.F.R. Sciences de l'Homme et de la Société
Master MIASHS – DCISS

25.10.2019

ALGORITHMIQUE

Devoir Surveillé

Durée : 2 heures – Tous documents autorisés, tout matériel électronique interdit.

Il sera tenu compte de la lisibilité des algorithmes. Pour chaque algorithme réalisé, on définira soigneusement les variables, les fonctions et les actions introduites pour résoudre le problème.
Les exercices sont indépendants et peuvent être traités dans n'importe quel ordre.

Dans un exercice, pour résoudre une question, vous pouvez utiliser une action ou une fonction que vous n'avez pas réussi à écrire, en donnant simplement sa spécification. Le barème donné est indicatif, il est proportionnel au temps de travail estimé : 1 points = 6 minutes.

Exercice 1 : Algorithme mystérieux (4 points)

Vous trouvez l'algorithme ci-dessous, composé d'une action nommée et d'un algorithme principal, écrit par un inconnu qui n'a donné aucune explication.

Après avoir compris le fonctionnement de l'action bricoler, donnez la trace d'exécution de l'algorithme, c'est-à-dire tous les messages qui seront affichés à l'écran pendant l'exécution, dans leur ordre d'affichage.

```
Lexique partagé
  e : écran
  z : entier
lexique principal
  a, b : entiers
  action utilisée : bricoler (modifié x : entier, élaboré y : entier)

algorithme principal
        e.afficher ("début")
        a ← 2 ; z ← 3
        e.afficher ("a = ", a , " b = " , b, ", z = ", z)
        bricoler (a, b)
        e.afficher ("a = ", a , " b = " , b, ", z = ", z)
        bricoler (b, z)
        e.afficher ("a = ", a , " b = " , b, ", z = ", z)
        e.afficher ("fin")
fin algorithme principal

action bricoler (modifié x : entier, élaboré y : entier)
lexique de bricoler
        k : entier
algorithme de bricoler
        e.afficher ("début de bricoler x= ", x, ", y = ", y, ", z = ", z, ", k = ",k)
        k ← x - z ; x ← x + 1
        y ← k + 5
        e.afficher ("fin de bricoler x= ", x, ", y = ", y, ", z = ", z, ", k = ",k)

fin algorithme de bricoler
```

Exercice 2 : Fonction mystérieuse (4 points)

On considère la fonction **bidule** suivante (qui ne respecte aucun schéma !) :

```
fonction bidule (x : entier) ⟶ entier
// bidule(x) renvoie ??? (à compléter)

Lexique de bidule
  j : entier
  trouvé : booléen

Algorithme de bidule
 j ← x
 trouvé ← faux
 tantque (j ≤ 12) et non trouvé faire
   si  j = 12
   alors
       trouvé ←vrai
   sinon
       j ←j + 1
   fsi
 ftq
 renvoyer(j)
```

a) Indiquez la condition d'arrêt de l'itération

b) En déduire la spécification de la fonction bidule : que renvoie cette fonction ?

c) Remplacez cette fonction par une fonction plus simple qui correspond à la spécification.

Exercice 3 : Fonction sur une chaine (4 points)

Ecrire la fonction **distanceCar** qui détermine la distance séparant la première occurrence de deux caractères donnés, dans une chaine donnée :

fonction **distanceCar** (c1, c2 : caractère, x : chaine) ⟶ entier ≥ 0
// distanceCar(c1, c2, x) renvoie la différence entre les positions de la première occurrence de c1 et
// la première occurrence de c2 dans la chaine x, renvoie 0 si c1 ou c2 n'est pas présent dans x
Exemples :
distanceCar('b','a',"charabia") renvoie 3 *car le 1er 'a' est en position 2 et le 1er b en position 5*
distanceCar ('a','h',"charabia") renvoie 1 car *le 1er 'h' est en position 1 et le 1er a en position 2*
distanceCar ('a','a',"charabia") renvoie 0 car c1 et c2 sont identiques
distanceCar ('d','a',"charabia") renvoie 0 car 'd' n'est pas présent dans "charabia"

Exercice 4 : Du pic à la forêt de sapins (8 points)

Un **pic** est une figure géométrique formée d'un sommet et de deux côtés égaux partant de ce sommet. Un angle caractérise l'écartement entre les 2 côtés du pic, un second angle définit l'orientation du premier côté du pic. (voir figure 1). On appelle Pic le type caractérisant n'importe quel pic traçable à l'aide d'une machine-tracés. Il est défini dans le lexique partagé de la manière suivante :

Pic : type agrégat s : Point, a : Angle, d : Angle, lg réel > 0 fagrégat

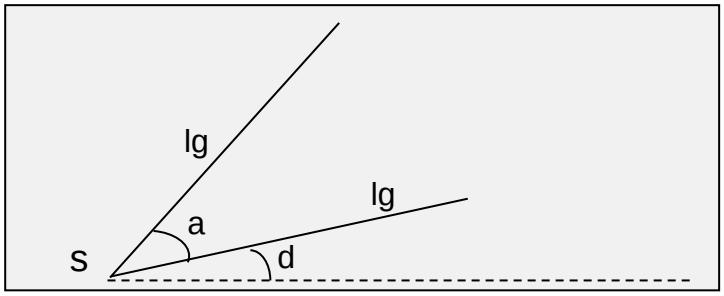

Figure 1 : caractérisation d'un pic

a) Réaliser l'action tracerPic qui trace le pic p à l'aide de la machine-tracés m :

tracerPic : action (consulté p : Pic ; modifié m : machine-tracés)
// Effet : trace le pic p à l'aide de m
// e.i. : écran et plume indifférents, p = (s_0, a_0, d_0, lg_0)
// e.f : le pic p est tracé, pp = s_0, cap = d_0, pe = haute

b) On veut maintenant dessiner un sapin composé d'un tronc de hauteur **h** (une droite) et des branches représentées par des pics séparés entre eux d'une distance k selon la forme présentée figure 2. Les pics sont tracés à partir de la cime du sapin (sommet **c**), ont des côtés de longueur **lg** et un écartement d'angle a, ils sont symétriques par rapport au tronc et leur nombre dépend de la longueur du tronc.

Définir le type Sapin qui caractérise un sapin.

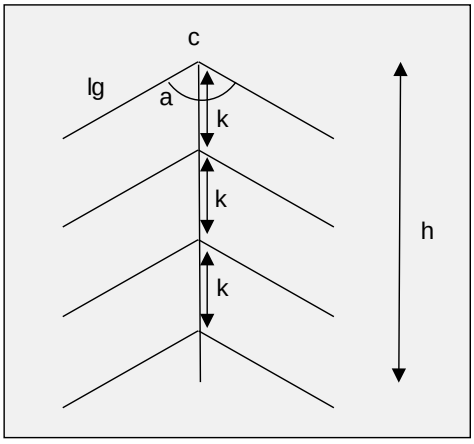

Figure 2 : caractérisation d'un sapin

c) Réaliser l'action tracerSapin qui trace le sapin s à l'aide de la machine-tracés m :

tracerSapin : action (consulté s : Sapin ;
modifié m : machine-tracés)
// Effet : trace le sapin s à l'aide de m
// e.i. : écran et plume indifférents
// e.f : le sapin s est tracé, ***état de la plume à définir***

d) Réaliser un algorithme principal qui lit les caractéristiques d'un sapin et les coordonnées de deux sommets, puis dessine une forêt formée de 3 sapins de même forme (voir figure 3 ci-dessous) :
- le premier sapin est le sapin lu, soit s1 ce sapin et s1.c la cime de ce sapin ;
- le second sapin sera tracé à partir du premier des deux sommets lus (c1), et aura une taille de 50% inférieure au premier sapin (branches, tronc et distance k divisées par deux)
- le troisième sapin sera tracé à partir du second des deux sommets lus (c2), et aura une taille de branches de 20% inférieure au premier sapin (branches de taille multipliée par 0.8) mais une longueur de tronc multipliée par 1.5 et la même distance k entre les pics que le premier sapin.

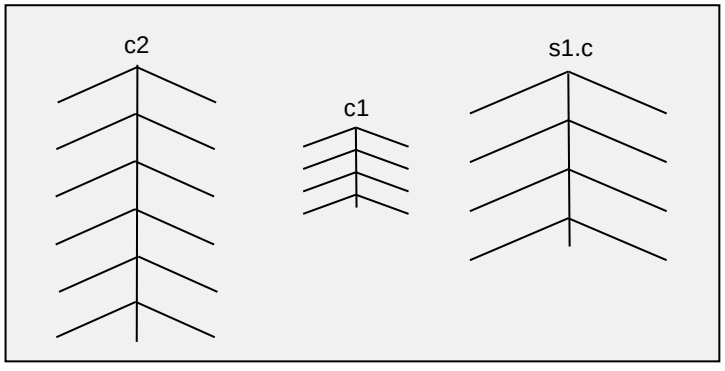

Figure 3 : une jolie forêt