CS110 – Introduction à l'algorithmique – Contrôle n^{o} 2

Aucun document autorisé
Nom : Durée : 25 minutes
Prénom : Barème indicatif sur 10 points

Quelques remarques générales :

- *il faut toujours donner la **spécification** de toute fonction ou procédure que vous rédigez, de préférence de manière claire et concise, sans forcément "copier-coller" l'énoncé.*
- *prenez soin d'indenter correctement les algorithmes.*
- *les déclarations de type utilisent '=' et pas ':'.*

De mauvaises (ou une absence de) spécifications ainsi qu'une mauvaise indentation seront sanctionnées à l'examen.

Soit la définition suivante :

```
type point = article
                x:réel
                y:réel
             fin article
```

2 points

Rédigez une fonction qui prend deux points en paramètre et retourne la distance entre ces deux points. On suppose qu'on dispose d'une fonction `racine` qui calcule la racine carrée d'un réel.

```
fonction distance(a,b:point):réel
(* calcule distance entre a et b *)
var dx, dy:réel
début
    dx ← b.x - a.x
    dy ← b.y - a.y
    retour( racine(dx×dx + dy×dy) )
fin
```

(1+)2+2 points

On dispose toujours de notre table traçante que l'on peut commander avec les trois mêmes procédures :
- `procédure position(x,y:réel)` déplace le crayon (sans écrire) à la position absolue (x,y).
- `procédure déplace(dx,dy:réel)` déplace le crayon (sans écrire) de dx unités sur l'axe des x et dy unités sur l'axe des y.
- `procédure trace(dx,dy:réel)` idem procédure `déplace` mais trace un trait pendant le déplacement.

1. Rédigez une procédure `tracepos` identique à `position` mais qui écrit pendant le déplacement.
   Cela est impossible... Il faudrait connaître la position actuelle. On pourrait par exemple imaginer que des fonctions `posx` et `posy` retournent l'abscisse et l'ordonnée courantes du crayon. On supposera toutefois que l'on dispose de cette procédure dans la suite.
   Ceux qui ont proposé des choses intéressantes ont eu 1/2 ou 1 point à cette question. Le dernier exercice a été noté sur 4 (2 et 2).
2. Rédigez une procédure qui prend trois points en paramètre et trace le triangle défini par ces trois points.

```
procédure trace_triangle(a,b,c:point)
(* trace le triangle (abc) *)
début
```

```
    position(a.x, a.y)
    tracepos(b.x, b.y)
    tracepos(c.x, c.y)
    tracepos(a.x, a.y)
fin
```

3. Proposez la définition d'un type `triangle` et rédigez une procédure traçant le triangle passé en paramètre.

```
type triangle = article
                   a,b,c:point
                fin article

procédure trace_triangle2(t:triangle)
(* trace le triangle t *)
début
    trace_triangle(t.a, t.b, t.c)
fin

une version plus longue acceptable était :

procédure trace_triangle2(t:triangle)
(* trace le triangle t *)
début
    position(t.a.x, t.a.y)
    tracepos(t.b.x, t.b.y)
    tracepos(t.c.x, t.c.y)
    tracepos(t.a.x, t.a.y)
fin
```

~~3~~ 2+2 points

1. Donnez une définition récursive (sur le modèle de la factorielle) de la fonction mathématique "puissance", qui associe à un réel x et un entier n la valeur x^n.
   x^0 = *1*
   $x^n = x \times x^{n-1}$ *si* $n \geq 1$ *(ne pas oublier cette condition sur n !)*
2. Rédigez une fonction récursive pour la fonction mathématique "puissance". Discutez sa terminaison.

```
fonction puissance(x:réel; n:entier):réel
(* calcule x puissance n (n ≥ 0) *)
début
    si n = 0 alors
          retour(1)
    sinon
          retour(x×puissance(x, n-1))
    fin si
fin
```

La fonction termine puisque chaque appel récursif se fait avec une valeur décrémentée pour n, on arrivera donc au cas de base n=0 au bout d'un nombre fini d'appels récursifs si n est positif. n négatif est une donnée invalide.