I3

Durée : *1h30*
Documents autorisés : **AUCUN** [1]

Remarques :
- Veuillez lire attentivement les questions avant de répondre.
- Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
- Rendez une copie propre.
- N'utilisez pas de crayon à papier sur votre copie.

1 Questions de cours : tri par insertion (5 points)

Nous avons vu en cours que l'algorithme du tri par insertion (tri en ordre croissant d'un tableau d'entiers) pouvait être :

procédure triParInsertion **(E/S** t :**Tableau**[1..MAX] **d'Entier**, **E** nb : **Naturel)**

Déclaration i,j : **Naturel**
temp : **Entier**

debut
pour i ←2 **à** nb **faire**
inserer(t,i)
finpour
fin

La particularité de cet algorithme est que la recherche de l'indice d'insertion et le décalage se font *en même temps* (procédure inserer).
- Donnez l'algorithme de la procédure insérer.
- Identifiez le meilleur des cas. Quelle est la complexité de l'algorithme dans ce cas ? Justifiez.
- Identifiez le pire des cas. Quelle est la complexité de l'algorithme dans ce cas ? Justifiez.

Solution proposée :

procédure inserer **(E/S** t : **Tableau**[1..MAX] **d'Entier**,**E** i : **NaturelNonNul)**

⌊**précondition(s)** i≤MAX

Déclaration temp : **Entier**
j : **NaturelNonNul**

debut
temp ← t[i]
j ← i-1
tant que t[j]>temp et j≥1 **faire**
t[j+1] ← t[j]
i ← j
j ← j-1
fintantque
t[i] ← temp
fin

- Dans le meilleur des cas (tableau déjà trié en ordre croissant), `insere` est en $\Omega(n) = 1$ donc `triParInsertion` en $\Omega(n) = n$
- ans le pire des cas (tableau déjà trié en ordre décroissant), `insere` est en $O(n) = n$ donc `triParInsertion` en $O(n) = n^2$

2 Approximation de π par la méthode de Monte-Carlo (15 points)

≪ Le terme méthode de Monte-Carlo, ou méthode Monte-Carlo, désigne toute méthode visant à calculer une valeur numérique en utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes.

1. Sauf les dictionnaires pour les étudiants non francophones

Si on tire aléatoirement un point $M(x, y)$ tel que $0 \leq x \leq 1$ et $0 \leq y \leq 1$, la probabilité que le point M appartienne au disque de centre O et de rayon 1 est de $\frac{\pi}{4}$.

En faisant le rapport du nombre de points dans le disque au nombre de tirages, on obtient une approximation du nombre $\frac{\pi}{4}$, donc de π, si le nombre de tirages est grand. (Cf. la figure 1) »(wikipédia)

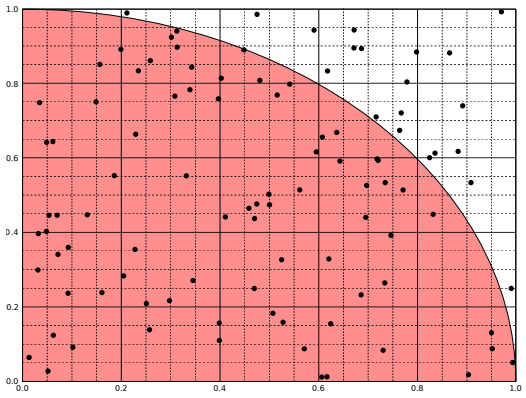


FIGURE 1 – Approximation de $\frac{\pi}{4}$ (wikipédia)

2.1 Analyse (5 points)

On considère posséder le type `Point2D` avec les opérations `point2D`, `abscisse`, `ordonnee`. On considère posséder aussi une opération `reelAleatoire` permettant d'obtenir un nombre réel aléatoire compris entre deux bornes réelles.

Complétez l'analyse descendante proposée par la figure 2, sachant que :

— chaque point du diagramme (en entrée et en sortie des opérations) représente un type à définir ;
— vous ne pouvez pas modifier le nombre d'entrées et de sorties de chaque opération ;
— l'opération `pointAleatoire` permet d'obtenir un point aléatoire dans une zone rectangulaire d'un plan ;
— l'opération `estDansCercle` permet de savoir si un point est à l'intérieur d'un cercle (défini par son centre et son rayon).

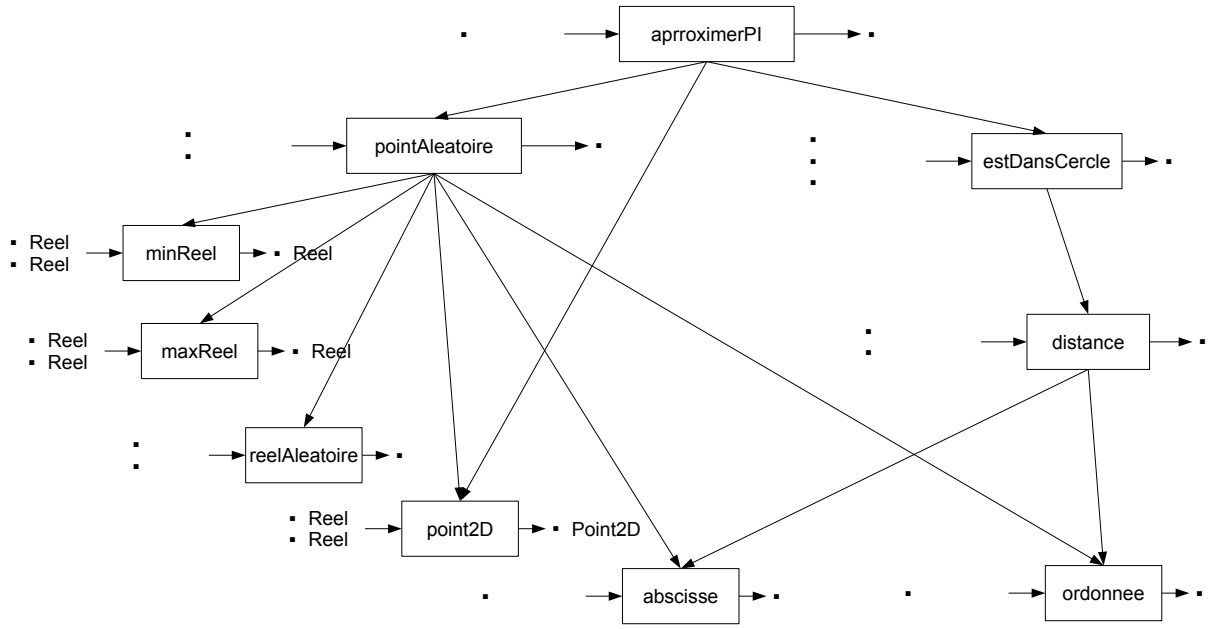
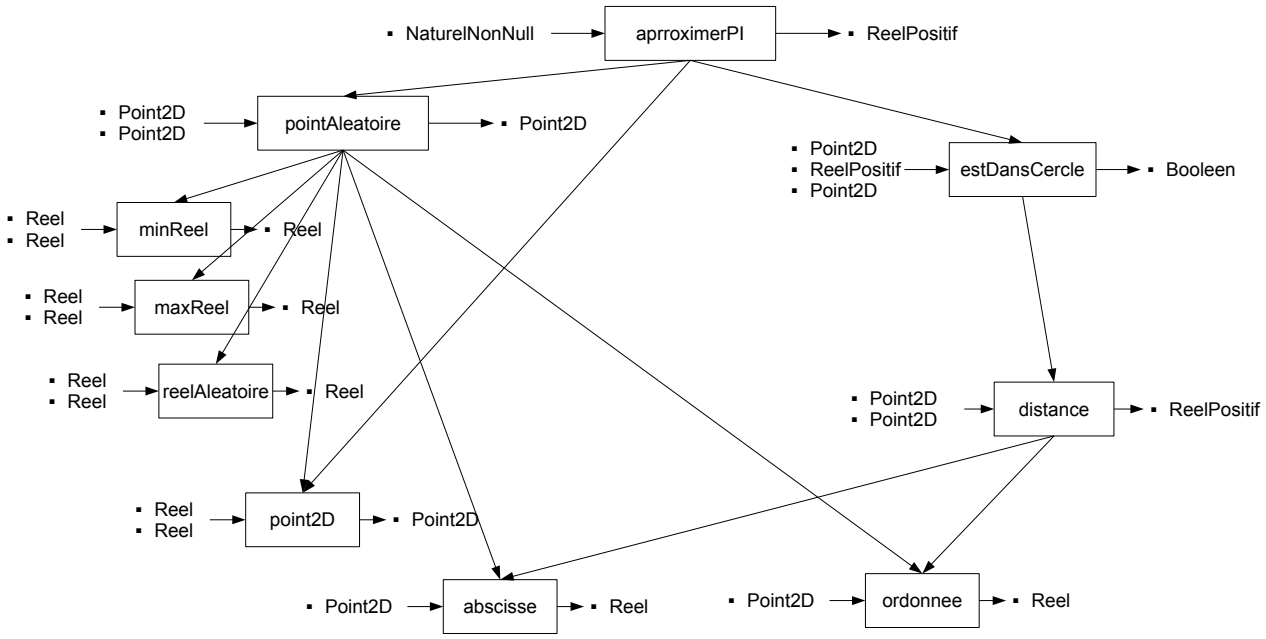


FIGURE 2 – Analyse descendante

Solution proposée :



2.2 Conception préliminaire (2 points)

Donnez les signatures des fonctions et procédures issues de l'analyse descendante.

Solution proposée :

fonction aproximerPI (nb : **NaturelNonNul**) **: ReelPositif**
fonction pointAleatoire (pt1, pt2 : Point2D) **:** Point2D

⌊**précondition(s)** pt1≠pt2

fonction estDansCercle (ptC : Point2D, r : **ReelPositif**, pt : Point2D) **: Booleen**
fonction reelAleatoire (min, max : **Reel**) **: Reel**

⌊**précondition(s)** min<max

fonction minReel (min, max : **Reel**) **: Reel**
fonction maxReel (min, max : **Reel**) **: Reel**
fonction abscisse (pt : Point2D) **: Reel**
fonction ordonnee (pt : Point2D) **: Reel**
fonction distance (pt1,pt2 : Point2D) **: ReelPositif**

2.3 Conception détaillée (8 points)

On représente le type `Point2D` de la façon suivante :
Type Point2D = **Structure**
x : Reel
y : Reel
finstructure
Donnez les algorithmes de toutes les fonctions et procédures de la conception préliminaire exceptées ceux des opérations `reelAleatoire`, `minReel` et `maxReel`.

Solution proposée :

fonction aproximerPI (nb : **NaturelNonNul**) **: ReelPositif**
debut
nbIn ← 0
pt0 ← point2D(0,0)
pt1 ← point2D(1,1)
pour i ←1 **à** nb **faire**
si estDansCercle(pt0,1,pointAleatoire(pt0,pt1)) **alors**
nbIn ← nbIn+1
finsi

```
    finpour
    retourner 4*nbIn/nb
fin
fonction pointAleatoire (pt1, pt2 : Point2D) : Point2D
    ⌊précondition(s)  pt1≠pt2
debut
    retourner point2D(reelAleatoire(minReel(abscisse(pt1),abscisse(pt2)), maxReel(abscisse(pt1),abscisse(pt2))),
    reelAleatoire(minReel(ordonnee(pt1),ordonnee(pt2)), maxReel(ordonnee(pt1),ordonnee(pt2))))
fin
fonction estDansCercle (ptC : Point2D, r : ReelPositif, pt : Point2D) : Booleen
debut
    retourner distance(ptC,pt)≤r
fin
fonction point2D (x,y : Reel) : Point2D
    Déclaration  resultat : Point2D
debut
    resultat.x ← x
    resultat.y ← y
    retourner resultat
fin
fonction abscisse (pt : Point2D) : Reel
debut
    retourner pt.x
fin
fonction ordonnee (pt : Point2D) : Reel
debut
    retourner pt.y
fin
fonction distance (pt1, pt2 : Point2D) : ReelPositif
    Déclaration  diffx,diffy : Reel
debut
    diffx ← abscisse(pt1)-abscisse(pt2)
    diffy ← ordonnee(pt1)-ordonnee(pt2)
    retourner sqrt(diffx*diffx+diffy*diffy)
fin
```