Cartouche du document

Année : ING 1
Matière : algo
Activité : Examen

Objectifs

Cet examen teste vos aptitudes en algorithmique
- à utiliser les types abstraits existants
- à définir des types abstraits
- à construire des algorithmes et en particulier à utiliser des structures de contrôles

Tous les documents (electronique et autres) sont autorisés.

La durée de l'examen est de 1h20 heures.

Sommaire des exercices

1 - Conteneur de nombres premiers : 6 points
2 - Fusion de listes : 6 points
3 - Un petit type abstrait : 6 points
4 - Réflexion : Comparaison de tris : 4 points

Corps des exercices

1 - Conteneur de nombres premiers : 6 points

Enoncé :

Nous allons écrire un algorithme permettant d'engendrer les nombres premiers inférieurs à une certaine valeur n entière passée en paramètre. Pour cela nous allons construire une structure linéaire initialisée par le nombre premier 2, à laquelle nous rajouterons les nouveaux nombres premiers en vérifiant qu'ils ne sont pas divisibles par les nombres premiers déjà trouvés.

Question 1)

Enoncé de la question

Quelle type abstrait choisiriez vous pour implémenter cet algorithme. Justifiez votre réponse.

Solution de la question

(1.5 pts)

On doit créer le conteneur avant de connaître le nombre exact d'éléments à mettre dedans. On choisira donc une liste plutôt qu'un vecteur. Les piles et les files ne sont pas retenues car dans

l'énoncé il n'est rien précisé sur l'utilisation du conteneur.

Question 2)

Enoncé de la question

Ecrire l'opération d'extension du type abstrait que vous avez choisi, qui engendre les nombres premiers inférieurs à un paramètre n selon la méthode ci-dessus.

Solution de la question

(4.5 pts)

L'opération *creerListeNombresPremiers*

2 - Fusion de listes : 6 points

Enoncé :

On souhaite fusionner deux listes quelconques L1 et L2 en une liste LF dans laquelle les éléments de L1 et L2 sont intercalés. Ainsi, LF contient le 1^{er} élément de L1, puis le 1^{er} élément de L2, puis le $2^{ème}$ élément de L1, puis le $2^{ème}$ élément de L2, etc. jusqu'à épuisement des éléments de L2. Le résiduel de L1 est ajouté à la fin de la liste LF.

On considèrera que la fusion ne peut pas être effectuée si la liste L1 est vide ou si le nombre d'éléments de la liste L2 est strictement supérieur à celui de L1.

Si L2 est vide, la fusion peut se faire et la liste fusionnée est identique à L1.

Question 1)

Enoncé de la question

Donner la signature exacte de cette opération d'extension. On n'oubliera pas de spécifier les préconditions de cette opération sous forme axiomatique.

Solution de la question

(1.5 points). Voir réponse globale.

Question 2)

Enoncé de la question

Ecrire l'opération d'extension.

Solution de la question

(4.5 points). Voir réponse globale.

Réponse globale à tout l'exercice

L'opération *intercalerListes*

3 - Un petit type abstrait : 6 points

Enoncé :

Il s'agit dans cet exercice de définir le type abstrait rectangle modélisant des rectangles dans l'espace R^2 . Dans cet espace, on ne s'intéresse qu'aux rectangles parallèles à l'axe des x.

On doit pouvoir construire un nouveau rectangle de deux façons différentes
- en donnant les coordonnées du point en haut à gauche et du point en bas à droite
- en donnant les coordonnées du point en haut à gauche, sa longueur et sa hauteur

Avec un objet de ce type, on doit pouvoir :
- récupérer le couple des coordonnées du point en haut à gauche
- récupérer le périmètre d'un rectangle
- tester la position d'un point par rapport à un rectangle (extérieure, intérieure ou sur la frontière)
- récupérer le couple des coordonnées du point en bas à droite
- récupérer la hauteur du triangle
- translater un rectangle
- récupérer la largeur du triangle

Dans cet exercice, on ne vous demande pas d'écrire les algorithmes des opérations d'extensions

Question 1)

Enoncé de la question

Donner la signature exacte des opérations de base. On n'oubliera pas de spécifier les préconditions (s'il y a lieu).

On jusfiera pourquoi elles ont été rangées dans la catégorie des opérations de base.

Solution de la question

(3 points). Voir réponse globale.

Question 2)

Enoncé de la question

Donner la signature exacte des opérations d'extension. On n'oubliera pas de spécifier les préconditions (s'il y a lieu).

On jusfiera pourquoi elles ont été rangées dans la catégorie des opérations d'extension.

Pour l'opération translater, on écrira sous forme axiomatique (postconditions) l'objet après transformation

Solution de la question

(3 points). Voir réponse globale.

Réponse globale à tout l'exercice

Il y a deux constructeurs. Ils sont équivalents. L'un des deux peut donc être une opération de base et

l'autre une opération d'extension car il peut être déduit de l'autre à l'aide d'un algorithme.

- Opérations de base :
  - récupérer le couple des coordonnées du point en haut à gauche
  - récupérer le couple des coordonnées du point en bas à droite
- Opérations d'extension :
  - récupérer le périmètre d'un rectangle
  - tester la position d'un point par rapport à un rectangle (extérieure, intérieure ou sur la frontière)
  - récupérer la hauteur du triangle
  - récupérer la largeur du triangle
  - translater un rectangle

Il y a plusieurs solutions possibles. Une solution est bonne dans la mesure où
- L'ensembles des opérations du type abstrait couvre le cahier de charges
- Toutes les opérations d'extension peuvent être déduites des opérations de base
- Il n'y a pas de redondance dans les opérations de base

4 - Réflexion : Comparaison de tris : 4 points

Enoncé :

En comparant les complexités des trois tris vus en travaux dirigés, donner les avantages et inconvénients de chacun d'eux.

Annexes (Arel)

```
Constructeur Liste : creerListeNombrePremiers(Entier  n) : Liste


Références locales
Liste l1, l2
Entier  val
Booleen  prem


// Début de l'algorithme
DEBUT
  l1  <-- listeVide()
  ajouter(l1,2)
  val  <--  3
  TANTQUE val <= n FAIRE // Structure  itérative
  DEBUT
```

```
    prem  <--  vrai
    l2  <--  l1
    TANTQUE NON estVide(l2) ET premier FAIRE // Structure  itérative
    DEBUT
      SI estEGal(val MOD premier(l2),0) ALORS  // Structure  conditionnelle
      DEBUT
        prem  <--  faux
      FIN
      SINON // Alternative  conditionnelle
      DEBUT
        l2  <--  reste(l2)
      FIN
    FIN
    SI prem ALORS  // Structure  conditionnelle
    DEBUT
      inserer(l1,longueur(l1) + 1, val)
    FIN
    val  <--  val + 2
  FIN
  retourner l1
FIN
```

```
definie(intercalerListe(l1,l2) ==> longueur(l1) >= longueur(l2)
Constructeur Liste : intercalerListe(Liste l1, Liste l2) : Liste


Références locales
Liste l3, l11, l21
Entier  val
Booleen  prem


// Début de l'algorithme
DEBUT
  l3  <-- listeVide()
  l11  <--  l1
  l21  <--  l2
  TANTQUE NON estVide(l2) FAIRE // Structure  itérative
  DEBUT
    ajouter(l3,premier(l11))
```

```
    ajouter(l3,premier(l21))
    l11  <--  reste(l11)
    l21  <--  reste(l21)
  FIN
  TANTQUE NON estVide(l1) FAIRE // Structure  itérative
  DEBUT
    ajouter(l3,premier(l11))
    l11  <--  reste(l11)
  FIN
  retourner l3
FIN
```