Algorithmique avancée et programmation C

Remise à Niveau

v2.3

N. Delestre

Table des matières

Le pseudo code

Vous écrirez vos algorithmes avec le pseudo code utilisé dans la plupart des cours d'algorithmique de l'INSA Rouen Normandie. Voici la syntaxe des instructions disponibles :

Type de données

Les types de base sont : **Entier**, **Naturel**, **NaturelNonNul**, **Reel**, **ReelPositif**, **ReelPositifNonNul**, **ReelNegatif**, **ReelNegatifNonNul**, **Booleen**, **Caractere**, **Chaine de caracteres**.

On définit un nouveau type de la façon suivante :
Type Identifiant_nouveau_type **=** Identifiant_type_existant

On déclare un tableau de la façon suivante :
— Tableau à une dimension : **Tableau[**borne_de_début. . .borne_de_fin**] de** type_des_éléments
— Tableau à deux dimensions : **Tableau[**borne_de_début. . .borne_de_fin**][**borne_de_début. . .borne_de_fin**] de** type_des_éléments
— . . .

On définit une structure de la façon suivante :
Type Identifiant **= Structure**
identifiant_attribut_1 : Type_1
. . .
finstructure

Affectation

Le symbole d'affectation est ←.

Conditionnelles

Il y a trois instructions conditionnelles :

si condition **alors**
instruction(s)
finsi

si condition **alors**
instruction(s)
sinon
instruction(s)
finsi

cas où identifiant_variable **vaut**
valeur_1:
instruction(s)_1
. . .
autre :
instruction(s)
fincas

Itérations

L'instruction de base pour les itérations déterministes est le **pour** :
pour identifiant ←borne_de_début **à** borne_de_fin **faire**
instruction(s)
finpour

On peut itérer sur les éléments d'une liste, d'une liste ordonnée ou d'un ensemble grâce à l'instruction **pour chaque** :
pour chaque élément **de** collection
instruction(s)

finpour

Pour les itérations indéterministes nous avons deux instructions :

tant que condition **faire**
instruction(s)
fintantque

repeter
instruction(s)
jusqu'a ce que condition

Sous-programmes

Les fonctions permettent de calculer un résultat :
fonction identifiant (paramètre(s)_formel(s)) **:** Type(s) de retour

⌊**précondition(s)** expression(s) booléenne(s)

Déclaration variable(s) locale(s)

debut
instruction(s) avec au moins une fois l'instruction **retourner**
fin

Les procédures permettent de créer de nouvelles instructions :
procédure identifiant **(**paramètre(s)_formel(s)_avec_passage_de_paramètres**)**

⌊**précondition(s)** expression(s) booléenne(s)

Déclaration variable(s) locale(s)

debut
instruction(s)
fin

Les passages de paramètre sont : entrée (**E**), sortie (**S**) et entrée/sortie (**E/S**).

1 Affectation et schéma conditionnel

1.1 Échanger

Ecrire une procédure, échangerDeuxEntiers, qui permet d'échanger les valeurs de deux entiers.

1.2 Parité

Écrire une fonction booléenne, estPair, qui à partir d'un nombre entier strictement positif retourne VRAI si ce nombre est pair et FAUX sinon.

1.3 Tri de trois entiers

Écrire une procédure, trierTroisEntiers, qui prend en entrée trois paramètres a, b et c contenant chacun un entier et qui les retourne triés par ordre croissant : a contient la valeur minimum, et c contient la valeur maximum.

1.4 Nombre de jours de congés

Dans une entreprise, le calcul des jours de congés payés s'effectue de la manière suivante : si une personne est entrée dans l'entreprise depuis moins d'un an, elle a droit à deux jours de congés par mois de présence (au minimum 1 mois), sinon à 28 jours au moins. Si cette personne est un

cadre et si elle est âgée d'au moins 35 ans et si son ancienneté est supérieure à 3 ans, il lui est accordé 2 jours supplémentaires. Si elle est cadre et si elle est âgée d'au moins 45 ans et si son ancienneté est supérieure à 5 ans, il lui est accordé 4 jours supplémentaires, en plus des 2 accordés pour plus de 35 ans.

Écrire une fonction, nbJoursDeConges, qui calcule le nombre de jours de congés à partir de l'âge exprimé en année, l'ancienneté exprimée en mois et l'appartenance (booléenne) au collège cadre d'une personne.

2 Schéma itératif

2.1 La multiplication

Écrire une fonction, multiplication, qui effectue la multiplication de deux entiers positifs (notés x et y) donnés en utilisant uniquement l'addition entière.

2.2 Calcul de factorielle n

Écrire une fonction, factorielle, qui calcule pour un entier positif donné n la valeur de n !.

2.3 Partie entière inférieure de la racine carrée d'un nombre

Écrire une fonction, racineEntiere, qui retourne la partie entière de la racine carrée d'un entier positif donné n (sans utiliser racineCarree).

2.4 Multiplication égyptienne

Les égyptiens de l'antiquité savaient :
— additionner deux entiers strictement positifs,
— soustraire 1 à un entier strictement positif,
— multiplier par 1 et 2 tout entier strictement positif,
— diviser par 2 un entier strictement positif pair.

Voici un exemple qui multiplie 14 par 13 en utilisant uniquement ces opérations :

14×13	= 14 + **14** $\times$ **(13 - 1)**	= 14 + **14** $\times$ **12**
	= 14 + **(14** $\times$ **2)** $\times$ **(12 / 2)**	= 14 + **28** $\times$ **6**
	= 14 + **(28** $\times$ **2)** $\times$ **(6 / 2)**	= 14 + **56** $\times$ **3**
	= 14 + 56 + **56** $\times$ **(3 - 1)**	= 70 + **56** $\times$ **2**
	= 70 + **(56** $\times$ **2)** $\times$ **(2 / 2)**	= 70 + **112** $\times$ **1**
	= 70 + 112	= 182

Donner le corps de la fonction suivante :

fonction multiplicationEgyptienne (a,b : **Naturel) : Naturel**

2.5 Intégration par la méthode des trapèzes

Écrire une fonction, integrale, qui retourne la valeur de l'intégrale d'une fonction f(x) réelle continue sur l'intervalle réel $[a, b]$. L'intégration consiste à découper cet intervalle, en n sous-intervalles de longueur Δ. L'intégrale d'un sous-intervalle [x, x + Δ] est approximée au trapèze de base Δ et de côtés f(x) et f($x + \Delta$). a, b et n vous sont donnés.

Remarque : la communication de f entre l'appelant et la fonction appelée, est réalisée de manière implicite (opération transparente pour vous). Cette remarque est valide pour tous les exercices numériques de ce type.

2.6 Nombres premiers

Écrire une fonction booléenne, estPremier, qui à partir d'un entier strictement positif donné, retourne le résultat VRAI ou FAUX selon que le nombre est premier ou non. Pour mémoire, voici la liste des nombres premiers inférieurs à 100 : 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97.

2.7 Recherche du zéro d'une fonction par dichotomie

Écrire une fonction, zeroFonction, qui calcule le zéro d'une fonction réelle f(x) sur l'intervalle réel [a, b], avec une précision epsilon. La fonction f ne s'annule qu'une seule et unique fois sur [a, b]. Pour trouver ce zéro, la recherche procède par dichotomie, c'est-à-dire divise l'intervalle de recherche par deux à chaque étape. Soit m le milieu de [a, b]. Si f(m) et f(a) sont de même signe, le zéro recherché est dans l'intervalle [m, b], sinon il est dans l'intervalle [a, m]. a, b et epsilon vous sont donnés.

3 Analyse descendante

3.1 Nombre de chiffres pairs dans un nombre

On se propose de calculer le nombre de chiffres pairs d'un nombre donné. On suit pour cela l'analyse descendante présentée par la figure 1, tel que :

nbChiffresPairs résoud le problème demandé. Par exemple pour le nombre 821, on obtient 2.

nbChiffres permet d'obtenir le nombre de chiffres d'un nombre. Par exemple pour le nombre 821, on obtient 3.

iemeChiffre permet d'obtenir le ième chiffre d'un nombre. Par exemple pour 821, le premier chiffre est 1, le second 2 et le troisième est 8 (on ne traite pas les erreurs).

estPair permet de savoir si un nombre est pair.

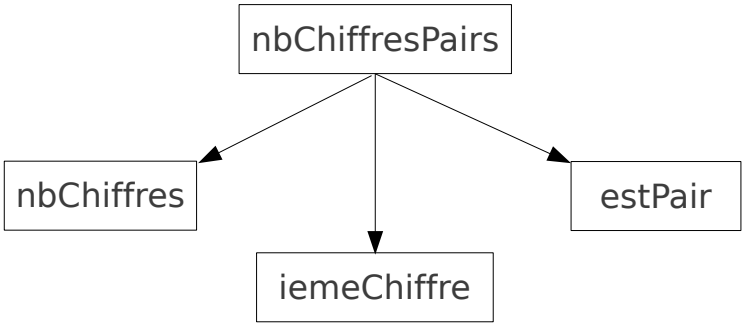

FIGURE 1 – Analyse descendante

1. Reprenez le schéma donné et complétez le (tel que nous l'avons vu en cours).
2. Donnez la signature des fonctions ou procédures des opérations données par l'analyse descendante.
3. Donnez le corps de la fonction ou procédure `nbChiffresPairs`.

3.2 Majuscule

La fonction $majuscule$ permet de calculer à partir d'une chaîne de caractères ch une chaîne de caractères ch' tel que tous les caractères minuscules, et uniquement eux, de ch soient transformés en majuscule dans ch'. La signature de cette est fonction est :
— **fonction** majuscule (uneChaine : **Chaine de caracteres) : Chaine de caracteres**
Ainsi $majuscule(" abc, ?ABC")$ donne la valeur $" ABC, ?ABC"$.

L'objectif de cet exercice est de donner l'algorithme de cette fonction en considérant que nous avons les trois fonctions suivantes :
— **fonction** longueur (uneChaine : **Chaine de caracteres) : Naturel**
— **fonction** iemeCaractere (uneChaine : **Chaine de caracteres**, position : **Naturel) : Caractere**
— **fonction** caractereEnChaine (unCaractere : **Caractere) : Chaine de caracteres**

3.2.1 Analyse

Pour calculer la version majuscule d'une chaîne de caractères ch, on a besoin de savoir calculer la majuscule d'un caractère c de ch lorsque c représente une lettre minuscule. Nous n'avons aucun a priori concernant la table de codage de ces caractères, si ce n'est que :
— le caractère 'a' précède le caractère 'b', qui précède le caractère 'c', etc.
— le caractère 'A' précède le caractère 'B', qui précède le caractère 'C', etc.

Proposez une analyse descendante de ce problème à l'aide du formalisme vu en cours.

3.2.2 Conception préliminaire

Déterminez la signature des fonctions ou procédures correspondant aux opérations de votre analyse descendante.

3.2.3 Conception détaillée

Donnez le corps de chacune de ces fonctions ou procédures.

3.3 Approximation de π par la méthode de Monte-Carlo

≪ 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.

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 2) ≫(wikipédia)

3.3.1 Analyse

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 3, 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 ;

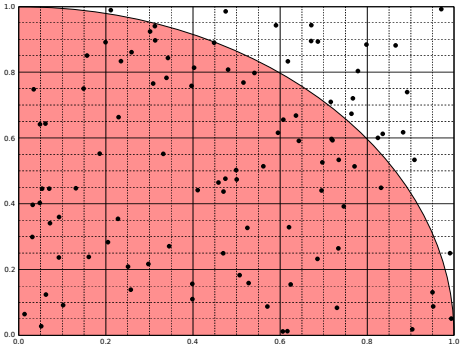

FIGURE 2 – Approximation de $\frac{\pi}{4}$ (wikipédia)

— 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).

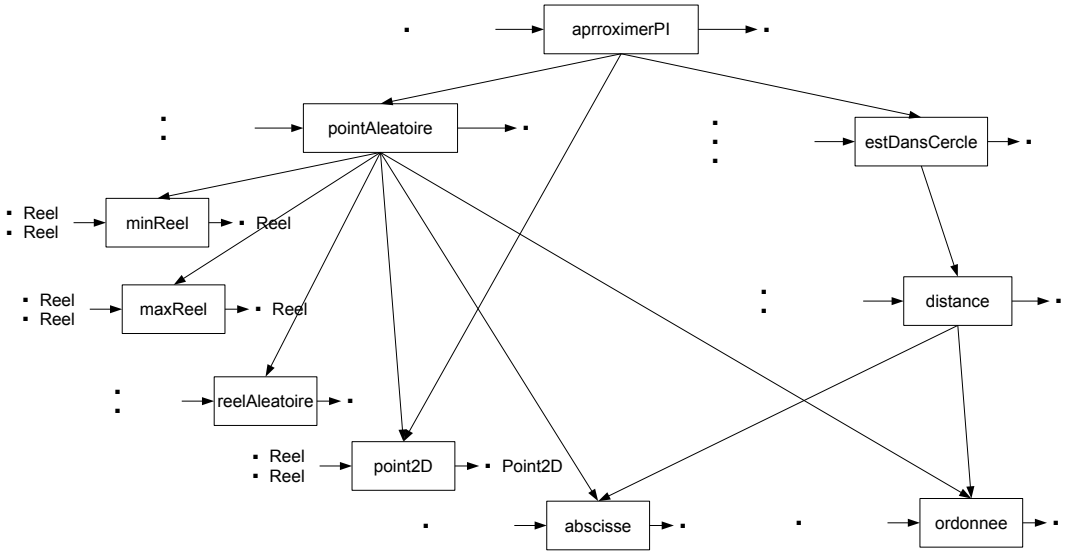

FIGURE 3 – Analyse descendante

3.3.2 Conception préliminaire

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

3.3.3 Conception détaillée

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`.

4 Tableaux

Dans tous les exercices qui vont suivre, le tableau d'entiers t est défini comme étant de type **Tableau[**1..MAX**] d'Entier**.

4.1 Plus petit élément d'un tableau d'entiers

Écrire une fonction, minTableau, qui à partir d'un tableau d'entiers t non trié de n éléments significatifs retourne le plus petit élément du tableau.

4.2 Indice du plus petit élément d'un tableau d'entiers

Écrire une fonction, indiceMin, qui retourne l'indice du plus petit élément d'un tableau d'entiers t non trié de n éléments significatifs.

4.3 Nombre d'occurrences d'un élément

Écrire une fonction, nbOccurences, qui indique le nombre de fois où un élément apparaît dans un tableau d'entiers t non trié de n éléments.

4.4 Recherche dichotomique

Écrire une fonction, rechercheDichotomique, qui détermine par dichotomie le plus petit indice d'un élément, (dont on est sûr de l'existence) dans un tableau d'entiers t trié dans l'ordre croissant de n éléments significatifs. Il peut y avoir des doubles (ou plus) dans le tableau.

5 Récursivité

5.1 Calcul de C(n,p)

Écrire une fonction `cnp`, qui en fonction des entiers positifs n et p, retourne le nombre de combinaisons de p éléments parmi n.

Pour rappel :

$$C_p^n = \begin{cases} 1 & \text{si } p = 0 \text{ ou } n = p \\ C_p^{n-1} + C_{p-1}^{n-1} & \end{cases}$$

5.2 Multiplication égyptienne

Soit la multiplication égyptienne définie dans l'exercice 2.4. On se propose cette fois d'écrire cet algorithme de manière récursive.

1. Déterminer le ou les cas d'arrêt (particuliers). Déterminer le ou les cas généraux.
2. Donner le corps de la fonction suivante en utilisant un algorithme récursif :

   fonction multiplicationEgyptienne (a,b : **Naturel) : Naturel**

5.3 Des cercles

Supposons que la procédure suivante permette de dessiner un cercle sur un graphique (variable de type `Graphique`) :

— **procédure** cercle **(E/S** g : Graphique,**E** xCentre,yCentre,rayon : **Reel)**

5.3.1 Compréhension

Soit l'algorithme suivant [1] :

procédure cercles **(E/S** g : Graphique,**E** x,y,r : **Reel**, n : **Naturel)**

Déclaration rTemp : **Reel**

debut

si n>0 **alors**

rTemp ← r/(1+racineCarree(2))
cercles(g,x,y+rTemp*racineCarree(2),rTemp,n-1)
cercles(g,x,y-rTemp*racineCarree(2),rTemp,n-1)
cercle(g,x,y,r)
cercles(g,x+rTemp*racineCarree(2),y,rTemp,n-1)
cercles(g,x-rTemp*racineCarree(2),y,rTemp,n-1)

finsi

fin

L'instruction `cercles(g,1.0,1.0,1.0,3)` permet d'obtenir le graphique de la figure 4.

Numérotez les cercles (numéro à mettre au centre du cercle) suivant leur ordre d'apparition sur le graphique.

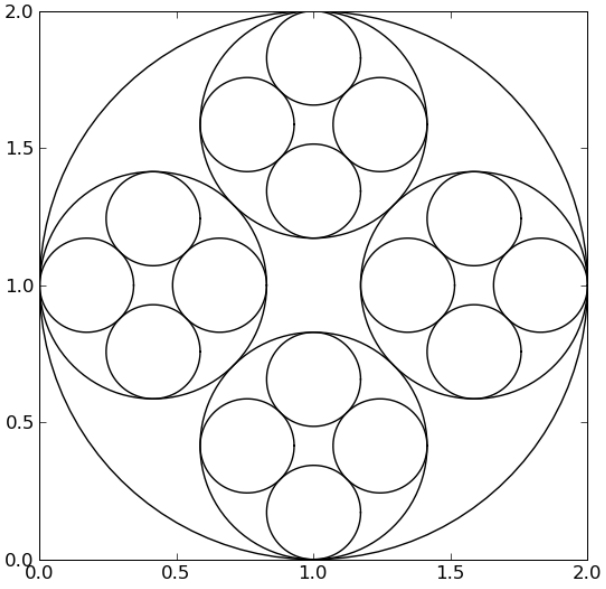

FIGURE 4 – Résultat d'un autre algorithme pour `cercles`

1. Pour comprendre les formules mathématiques de cet algorithme, il faut considérer le carré définit par les 4 centres des cercles, de rayon r', intérieurs au cercle courant, de rayon r. Son côté est de $2r'$. Les centres sont donc a une distance de $r'\sqrt{2}$ du centre du cercle courant et donc $r = r'\sqrt{2} + r'$

5.3.2 Construction

Proposez un autre algorithme de la procédure `cercles` qui, pour la même instruction `cercles(g,1.0, 1.0,1.0,3)`, affiche les cercles dans l'ordre proposé par la figure 5.

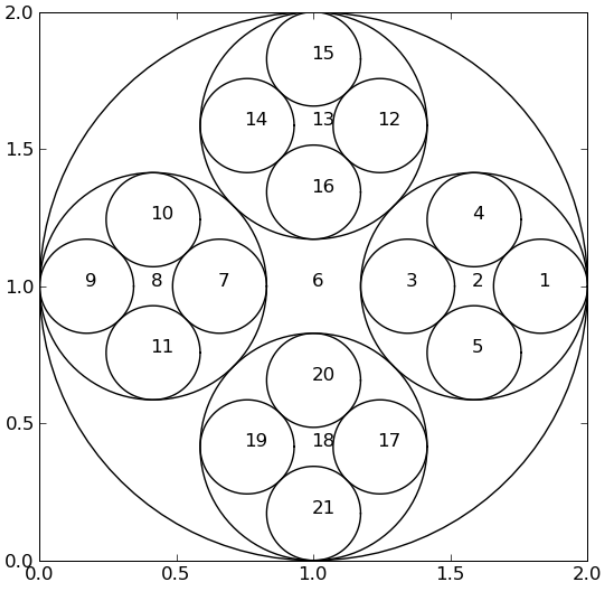

FIGURE 5 – Résultat d'un autre algorithme pour `cercles`

5.4 Des étoiles

Rappels mathématiques

Les trois sommets d'un triangle équilatéral, dont l'un des côté est parallèle à l'axe des abscisses, de centre x_c, y_c de base b ont les coordonnées suivantes :
— $x_c, y_c + 2 * h/3$
— $x_c - b/2, y_c - h/3$
— $x_c + b/2, y_c - h/3$
avec $h = \sqrt{b^2 - (b/2)^2}$

Questions

Supposons que la procédure suivante permette de dessiner un segment sur un graphique (variable de type `Graphique`) :
— **procédure** ligne **(E/S** g : Graphique,**E** x1,y1,x2,y2 : **Reel)**

L'objectif est de concevoir une procédure `dessinerCroix` qui permet de dessiner sur un graphique des dessins récursifs tels que présentés par la figure 6. La signature de cette procédure est :
— **procédure** dessinerCroix **(E/S** g : Graphique,**E** xCentre,yCentre,base : **Reel**, niveauRecursion : **Naturel)**

1. Lors du premier appel de cette procédure, donnez la valeur des quatre derniers paramètres effectifs afin d'obtenir le graphique de la figure 6.

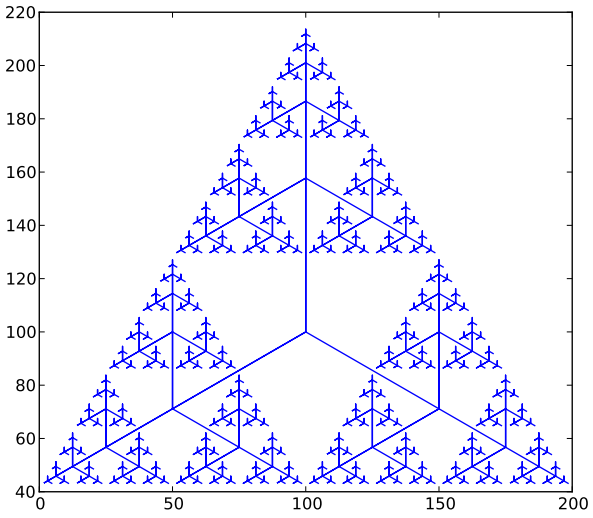

FIGURE 6 – Dessin récursif

2. Donnez le corps de cette procédure.

6 Tris

6.1 Tri par insertion

Nous avons vu en cours que l'algorithme du tri par insertion est :

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**
j ← obtenirIndiceDInsertion(t,i,t[i])
temp ← t[i]
decaler(t,j,i)
t[j] ← temp
finpour
fin

Donnez l'algorithme de la fonction obtenirIndiceDInsertion tout d'abord de manière séquentielle, puis de manière dichotomique. Démontrez que la complexité de ce dernier est-il en $O(log_2(n))$.

6.2 Tri shaker

La tri shaker est une amélioration du tri à bulles où les itérations permettant de savoir si le tableau et trié (et qui inverse deux éléments successifs $t[i]$ et $t[i+1]$ lorsque $t[i] > t[i+1]$) se font successivement de gauche à droite puis de droite à gauche.

Donnez l'algorithme du tri shaker.

7 Structure dynamique de données `ListeChaineeDEntiers`

Soit le type `ListeChaineeDEntiers` défini de la façon suivante :
Type ListeChaineeDEntiers **=** ^ NoeudDEntier
Type NoeudDEntier **= Structure**
entier : **Entier**
listeSuivante : ListeChaineeDEntiers
finstructure
Le principe d'encapsultation nous incite à utiliser ce type à l'aide des fonctions et procédures :

— **fonction** listeVide **() :** ListeChaineeDEntiers
— **fonction** estVide **(**uneListe : ListeChaineeDEntiers**) : Booleen**
— **procédure** ajouter **(E/S** uneListe : ListeChaineeDEntiers,**E** element : Entier**)**
— **fonction** obtenirEntier **(**uneListe : ListeChaineeDEntiers**) :** Entier
⌊**précondition(s)** $non(estVide(uneListe))$
— **fonction** obtenirListeSuivante **(**uneListe : ListeChaineeDEntiers**) :** ListeChaineeDEntiers
⌊**précondition(s)** $non(estVide(uneListe))$
— **procédure** fixerListeSuivante **(E/S** uneListe : ListeChaineeDEntiers,**E** nelleSuite : ListeChaineeDEntiers**)**
⌊**précondition(s)** $non(estVide(uneListe))$
— **procédure** supprimerTete **(E/S** l :ListeChaineeDEntiers**)**
⌊**précondition(s)** non estVide(l)
— **procédure** supprimer **(E/S** uneListe : ListeChaineeDEntiers**)**

7.1 Conception

Donnez le corps de la procédure `ajouter`.

7.2 Utilisation

1. Proposez une procédure qui affiche tous les entiers d'une liste chaînées d'entier.
2. Proposez une fonction itérative qui permet de savoir si un entier est présent dans une liste chaînées d'entiers.
3. Proposez une fonction récursive qui permet de savoir si un entier est présent dans une liste chaînées d'entiers.