Algorithmes et Structures de Données
Mardi 6 Novembre 2018

Durée 1H30 – Cours et TD **NON** autorisés

Exercice 1 : Fichiers textes et tableaux (8 pts)

1.1. Expliquer la méthode en français et écrire en pseudo-langage les structures de données nécessaires.

On ne doit lire qu'une seule fois chaque fichier et utiliser un tableau de chaines pour sauvegarder les mots lus. Donc, on lit chaque fichier ligne à ligne et pour chaque ligne lue, on extrait les mots (grâce à mot-suiv). On vérifie que le mot n'appartient pas déjà au tableau et on l'ajoute. Une fois les les mots enregistrés dans le tableau, on les écrits dans un nouveau fichier ligne par ligne où chaque ligne est une chaine formée d'un groupe de 10 mots. La dernière ligne sera incomplète.

1.2. Ecrire un programme en pseudo-langage (comprenant plusieurs procédures et/ou fonctions) qui produit le nouveau fichier. On pourra utiliser la procédure `mot-suiv(c,i,m)` vue en TD.

```
Programme fichier-mots
Const max = 500
Type tab = tableau[1..max] de chaine
Var tb : tab
    nb : entier

Fonction appartient-tab(t : tab, n : entier, m : chaine) : booléen
Var i : entier
Début
i←1
TantQue (i<=n) et (t[i]<>m) faire
      i←i+1
FintantQue
Retourner (i<=n)
Fin

Procédure remplir-tab(E nf : chaine , E/S t : tab, n : entier)
Var c, m : chaine
    i : entier
    f : FT
Début
f←Ouvrir-en-lecture(nf)
TantQue ¬finfichier(f) Faire
      c←lireChaine(f)
      i←1
      TantQue i<=lg(c) Faire
          mot-suiv(c,i,m)
          Si m<>''
             Alors Si (n=0) ou (¬appartient-tab(t,n,m))
                        Alors n←n+1
                              t[n]←m
                   FinSi
          FinSi
      FinTantque
FinTantQue
Fermer(f)
Fin
```



```
Procédure écrit-fich(E nf : chaine , t : tab, n : entier)
Var f : FT
    C : chaine
    i : entier
Début
f←ouvrir-en-écriture(nf)
c←''
Pour i←1 à n inc +1 faire
   c←concat(c,t[i],' ')
   Si (i mod 10)=0 ou (i=n)
       Alors écrire-chaine(f,c)
             c←''
   FinSi
FinPour
Fermer(f)
Fin

Début
nb←0
remplir-tab('mots1.txt',tb,nb)
remplir-tab('mots2.txt',tb,nb)
ecrit-fich('mots3.txt',tb,nb)
Fin
```

Exercice 2 : Pile (6 pts)

2.1. Ecrire en pseudo-langage la fonction `racine(x,i)`. Ne pas oublier de numéroter les appels récursifs.

```
Fonction racine(x,i : entier) : réel
Var res : réel
Début
Si i=0
   Alors res←x/2
   Sinon res←(racine(x,i-1)+x/racine(x,i-1))/2
FinSi
Retourner(res)
Fin
```

2.2. Simuler la pile pour l'appel `écrire(racine(5,3)){@0}` dans le programme principal.

15 appels
res = 161/144+180/161 = 2,2360

~~@2, x=5, i=0~~	res=5/2
~~@1, x=5, i=0~~	res=5/2
~~@2, x=5, i=1~~	res=(5/2+(5/(5/2)))/2=(5/2+2)/2=9/4
~~@2, x=5, i=0~~	res=5/2
~~@1, x=5, i=0~~	res=5/2
~~@1, x=5, i=1~~	res=(5/2+(5/(5/2)))/2=(5/2+2)/2=9/4
~~@2, x=5, i=2~~	res=1/2*(9/4+5/(9/4))=9/8+10/9=161/72
~~@2, x=5, i=0~~	res=5/2
~~@1, x=5, i=0~~	res=5/2
~~@2, x=5, i=1~~	res=(5/2+(5/(5/2)))/2=(5/2+2)/2=9/4
~~@2, x=5, i=0~~	res=5/2
~~@1, x=5, i=0~~	res=5/2
~~@1, x=5, i=1~~	res=(5/2+(5/(5/2)))/2=(5/2+2)/2=9/4
~~@1, x=5, i=2~~	res=1/2*(9/4+5/(9/4))=9/8+10/9=161/72
~~@0, x=5, i=3~~	res=1/2*(161/72+5/(161/72))=161/144+180/161=2,2360

Exercice 3 : Binaire (6 pts)

3.1. Ecrire les **fonctions itérative et récursive** qui retournent la notation décimale (un entier) d'un nombre binaire (donné dans une chaîne de caractères).

```
Fonction bin2int-it(c : chaine) : entier
Var res, i, j : entier
Début
j←lg(c)
res←0
Pour i←0 à j-1 inc +1 faire
   res ← res + char2int(c[j-i])*puiss(2,i)
FinPour
Retourner(res)
Fin

Fonction bin2int-rec(c : chaine, i : entier) : entier
Var res : entier
Début
Si i>lg(c)
   alors res ← 0
   sinon res ← char2int(c[lg(c)-i+1])*puiss(2,i-1) + bin2int-rec(c,i+1)
FinSi
Retourner(res)
Fin

Appel de bin2int-rec('11001',1))
```

3.2. Ecrire les **fonctions itérative et récursive** qui retournent la notation binaire (chaîne de caractères) d'un entier naturel.

```
Fonction int2bin-it(n : entier) : chaine
Var res : chaine
Début
res ← ''
répéter
   res ← concat(int2char(n mod 2),res)
   n ← n div 2
jusqu'à n=0
retourner(res)
Fin

Fonction int2bin-rec(n : entier) : chaine
Var res : chaine
Début
   Si n=0
      Alors res ← ''
      Sinon res ← concat(int2bin-rec(n div 2),int2char(n mod 2))
  FinSi
retourner(res)
Fin
```