Premier cycle Vendredi 11 Mai 2007

I3 - Algorithmique et Langage C

Durée : *1h30*
Documents autorisés : **AUCUN** (calculatrice comprise)

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.
- Justifiez vos réponses.

1 Développement limité (6 points)

Lorsque x est proche de 0, $sinh(x)$ peut être approximé à l'aide de la formule suivante :

$$\sum_{i=0}^{n} \frac{x^{2i+1}}{(2i+1)!}$$

Complétez le corps de la fonction suivante qui calcule une approximation de $sinh(x)$ jusqu'au rang n. Utilisez uniquement les variables et paramètres donnés et n'utilisez aucune autre fonction (pas d'analyse descendante) :

fonction sinh (x : **Réel**, n : **Naturel) : Réel**
  Déclaration i : **Naturel**
    numerateur,denominateur : **Réel**
    resultat : **Réel**
début
  ...
fin

Solution proposée :

fonction sinh (x : **Réel**, n : **Naturel) : Réel**
  Déclaration i : **Naturel**
    numerateur,denominateur : **Réel**
    resultat : **Réel**
début
  numerateur ← x
  denominateur ← 1
  resultat ← resultat+numerateur/denominateur
  pour i ←1 **à** n **faire**

```
        numerateur ← numerateur*x*x
        denominateur ← (2*i+1)*(2*i)*denominateur
        resultat ← resultat+numerateur/denominateur
    finpour
    retourner resultat
fin
```

2 Nombre de chiffres pairs dans un nombre (7 points)

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.

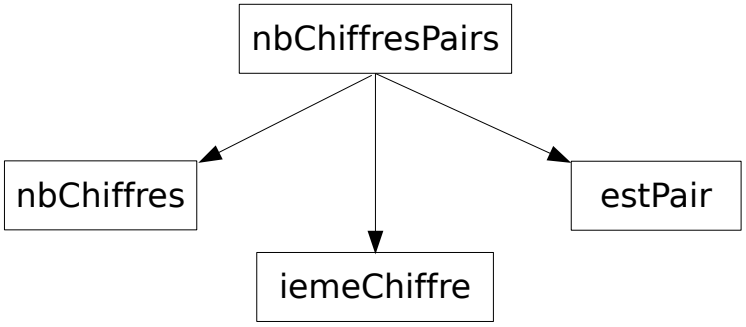
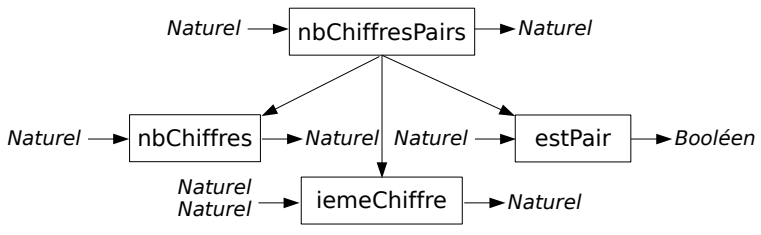


FIG. 1 – Analyse descendante

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`.

Solution proposée :

1.



2.
   - **fonction** nbChiffresPairs (nb : **Naturel) : Naturel**
   - **fonction** nbChiffres (nb : **Naturel) : Naturel**
   - **fonction** iemeChiffre (nb : **Naturel**,) **: Naturel**
   - **fonction** estPair (nb : **Naturel**,) **: Booléen**

3.
```
fonction nbChiffresPairs (nb : Naturel) : Naturel
   Déclaration  i : Naturel
                resultat : Naturel
début
   resultat ← 0
   pour i ←1 à nbChiffres(nb) faire
      si estPair(iemeChiffre(nb,i)) alors
         resultat ← resultat+1
      finsi
   finpour
   retourner resultat
fin
```

3 Tri par insertion (7 points)

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

```
fonction obtenirIndiceDInsertion (t : Tableau[1..MAX] d'Entier ,borneSup :Naturel,lEntier :Entier)
: Naturel

procédure decaler ( E/S t : Tableau[1..MAX] d'Entier  , E borneInf,borneSup :Naturel )

procédure triParInsertion ( E/S t : Tableau[1..MAX] d'Entier  , E nb :Naturel )
   Déclaration  i,j,temp : Naturel
début
   pour i ←2 à nb faire
      j ← obtenirIndiceDInsertion(t,i,t[i])
      temp ← t[i]
      decaler(t,j,i)
      t[j] ← temp
   finpour
fin
```

1. Donnez le corps de la fonction `obtenirIndiceDInsertion`.
2. Quelle est la complexité dans le pire des cas de la procédure `triParInsertion` ? Justifiez votre réponse.

Solution proposée :

1.

fonction obtenirIndiceDInsertion (t : **Tableau[**1..MAX**] d'Entier** ,borneSup :**Naturel**,lEntier :**Entier)** **: Naturel**

Déclaration i : **Naturel**

début

i $\leftarrow$ 1

tant que t[i]<lEntier et i<borneSup **faire**

i $\leftarrow$ i+1

fintantque

retourner i

fin

2. Complexité en $O(n^2)$, en effet :
   - La boucle principale est en $O(n)$ (car c'est une boucle `pour` en dont les bornes est fonction du nombre d'éléments). On va donc multiplier cette complexité par l'addition des complexités des instructions qu'elle contient :
     - `obtenirIndiceDInsertion` est en $O(n)$ (si recherche séquentiel, boucle pour fonction du nombre d'éléments) ou $(Olog_2(n)$ (si recherche dichotomique)
     - `decaler` est en $O(n)$ car boucle pour fonction du nombre d'éléments
   - Cette suite d'instruction est donc en $O(n)$ et donc la complexité de `triParInsertion` dans le pire des cas est en $O(n^2)$