Premier cycle Jeudi 22 Juin 2006

I3 - Algorithmique et Langage C

Correction

1 Multiplication égyptienne

- Cas terminal :
  - Si $b = 1$ alors $a * b = a$
- Cas général, Si $b > 1$
  - Si b est paire alors $a * b = 2a * b/2$
  - Si b est impaire alors $a * b = a + a * (b - 1)$

```
fonction multiplicationEgyptienne (a,b :Naturel) : Naturel
début
   si b=1 alors
      retourner a
   sinon
      si b mod 2=0 alors
         retourner multiplicationEgyptienne(2*a,b div 2)
      sinon
         retourner a+multiplicationEgyptienne(a,b-1)
      finsi
   finsi
fin
```

2 Drapeau tricolore

Puisqu'il faut trier, nous avons besoins d'une fonction de comparaison sur le type Couleur :

```
fonction estPlusPetitOuEgal (a,b : Couleur) : Booléen
début
   si a=Bleu ou (a=Rouge et b≠Bleu) ou (a=Blanc et b=Blanc) alors
      retourner Vrai
   sinon
      retourner Faux
   finsi
fin
```

Dans ce cas l'algorithme construireDrapeauFrancais est équivalent à l'un des algo de trie vu en cours en utilisant la fonction estPlusPetitOuEgal à la place de l'opérateur $<$

3 Les pointeurs en C

Soit p un pointeur qui "pointe" sur un tableau t :

```
int t[] = {12, 23, 43, 45, 56, 67, 79, 89, 90};
int *p;
p = t;
```

1. `p+(*p-10)` : Adresse du 3ème élément de t
2. `*(p+2)` : 43
3. `*p+2` : 14
4. `t+3` : Adresse du 4ème élément de t
5. `*(&(t[4])-3)` : 23
6. `*(p+*(p+8)-t[7])` : 23

4 Encore un peu de C

5 Et pour finir un peu d'algo (5 points)

5.1 Initialiser

Écrire le corps de la procédure suivante qui initialise (aucun élément) une variable de type Liste :

procédure initialiser **(Entrée/Sortie** l : Liste **)**
début
  l.nbElements ← 0
  l.indicePremierElement ← -1
fin

5.2 Parcourir

procédure afficher **(Entrée** l : Liste **)**
  Déclaration i : **Naturel**
début
  i ← l.indicePremierElement
  tant que i≠-1 **faire**
    écrire(lesElements[i])
    i ← l.suivant[i]
  fintantque
fin

5.3 Insérer

procédure insérer **(Entrée/Sortie** l : Liste , **Entrée** nom : **Chaîne de caractères)**
  Déclaration iprecedent,i : **Naturel**doitEtreInsereIci : **Booléen**
début
  i ← l.indicePremierElement
  iprecedent ← -1

```
    doitEtreInsereIci ← Faux
    tant que non doitEtreInsereIci et i≠-1 faire
        si estInferieurOuEgal(nom,l.lesElements[i]) alors
            doitEtreInsereIci ← Vrai
        sinon
            iprecedent ← i
            i ← suivant[i]
        finsi
    fintantque
    l.nbElements ← l.nbElements+1
    l.lesElements[l.nbElements] ← nom
    l.suivant[l.nbElements] ← i
    si iprecedent=-1 alors
        l.indicePremierElement ← l.nbElements
    sinon
        l.suivant[iprecedent] ← l.nbElements
    finsi
fin
```