Rapport
Mini projet 1 - Tri par distribution

Ken Hasselmann, Anne Jamet

22 décembre 2011

1. La structuration de l'environnement se fait à l'aide d'une matrice qui contiendra un tableau de 10 lignes et 100 colonnes afin de pouvoir stocker tous les nombres. Ainsi qu'une liste du nombre d'élements par colonnes.

```
#include <iostream>
#include <math.h>
#include <stdlib.h>

typedef struct matrice
{
int tab[10][100];
int elt[10];
} matrice;

void insert(int nb, matrice* mat, int etape) ;
void sort(matrice* in, matrice* out, int etape);
void afficher(matrice* mat);
void initialise(matrice* mat);
```

   On saisit les chiffres qui iront dans une fonction de tri initial, qui les placera dans le tableau, puis on fera le tri suivant les colonnes et ensuite on change d'indice de tri et on recommence le tri suivant les colonnes et cela jusqu'à atteindre le nombre d'étapes defini par le plus grand nombre.

2. La saisie des nombres se fait dans la fonction main, à l'aide de la fonction insert.

```
    cout << "Combien de nb à classer ?" << endl;
    cin >> nbelt;
    cout << "Entrez ces nombres :" << endl;
    for (int i(0);i<nbelt;i++)
    {
        cin >> nb;
        insert(nb, m1, 0);
```

3. Pour connaitre le nombre d'étapes on fait une boucle sur l'ensemble des nombres qui incrémentera une variable dès qu'il trouve un nombre ayant plus de chiffres que le précédent.

```
24                  while(nb > pow(10,n)) n++;
```

4. Pour déterminer à quelle classe appartient un nombre, on extrait le chiffre des unités, dizaines ou centaines (etc...) grâce à la partie entiere du nombre divisé par 1,10 ou 100 modulo 10.

```
68  void insert(int nb, matrice* mat, int etape)
69  //fonction d'insertion des nombres dans une matrice suivant une etape
70  {
71      int indice = (int)floor(nb/pow(10,etape))%10;
72      (*mat).tab[indice][(*mat).elt[indice]] = nb;
73      (*mat).elt[indice]++;
74  }
```

5. Grâce la structure du programme, il suffit de faire une boucle entre l'insertion dans une matrice, le tri et l'échange puis on échange les pointeurs des matrices pour recommencer l'opération.

```
30      for (int j(1);j<n;j++)
31      {
32          sort(m1, m2, j); //on tri
33          matrice* tmp = m1;
34          m1 = m2; //on echange les pointeurs des 2 matrices
35          m2 = tmp;
36          initialise(m2);
37   //on remet les nombres d'elements à 0 dans la matrice d'arrivée
38      }
39      cout << "Resultat :" << endl;
40      afficher(m1);
41      delete m1;    //On libère la case mémoire
42      m1 = 0;
43      delete m2;    //On libère la case mémoire
44      m2 = 0;
```