TD1: Manipulation des tableaux en C

Programmation en C (LC4)

Semaine du 28 janvier 2008

Nous modélisons des permutations de $[0 \dots n-1]$ dans $[0 \dots n-1]$ à l'aide d'un tableau. La valeur de la case i du tableau correspond à l'image de i par la bijection. Ainsi la bijection :

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 3 & 0 & 2 \end{pmatrix}$$

est dénotée par le tableau C :

```
int f[4] = { 1, 3, 0, 2 };
```

▶ Exercice 1

```
int est_identite (int n, int f[])
{
  int i = 0;
  for (i = 0; i < n; i++)
    if (f[i] != i)
      return 0;
  return 1;
}
```

▶ Exercice 2

```
int nombre_inversion (int n, int f[])
{
  int i, j;
  int acu = 0;
  // On n'explore que les couples (i, j) tels que i < j.
  for (i = 0; i < n; i++)
    for (j = i + 1; j < n; j++)
      if (f [i] > f [j])
        acu++;
  return acu;
}
```

▶ Exercice 3

```
void compose_permutation(int n, int perm1[],int perm2[], int perm_compose[])
{
  int i;
  for(i = 0; i < n; i++)
    perm_compose[i]=perm1[perm2[i]];
}
```

▶ **Exercice 4**

```
void permutation_inverse(int n, int perm[], int perm_inverse[])
{
  int i;
  for(i = 0; i < n; i++)
    perm_inverse[perm[i]]=i;
}
```

▶ **Exercice 5**

```
void permutation_aleatoire (int n, int f[])
{
  int i, j;
  int tmp;

  // On part de l'identite.
  for (i = 0; i < n; i++)
    f[i] = i;

  // On echange chaque element i avec un element tire au hasard.
  for (i = 0; i < n; i++)
  {
    j = entier_hasard (n-i)+i;

    tmp = f[i];
    f[i] = f[j];
    f[j] = tmp;
  }
}
```

▶ **Exercice 6**

```
void applique_permutation (int n, int perm[], int src[], int dst[])
{
  int i, tmp;
  for (i = 0; i < n; i++)
    dst[perm[i]] = src[i];
}
```

▶ **Exercice 7**

```
int est_trie (int n, int t[])
{
  int i = 0;
  for (i = 0; i < n - 1;i++)
  {
    if (t[i] > t[i + 1])
      return 0;
  }
  return 1;
}
```

► **Exercice 8**

```
void trier (int n, int non_trie[], int trie[],int perm[])
{
  do {
    permutation_aleatoire (n, perm);
    applique_permutation(n, perm, non_trie,trie);
  } while (est_trie (n, trie) == 0);

}
```

► **Exercice 9**

```
void genere_permutation_de_tri (int n, int t[],, int perm[])
{
  int i, j;
  for (i = 0; i < n; i++)
    perm[i] = 0;

  for (i = 0; i < n; i++)
    for (j = i + 1; j < n; j++)
      if (t[i] < t[j])
        perm[j]++;
      else
        perm[i]++;
}
```