Récursivité : le coup de grâce.
A propos de l'opération de mélange

Des langages comme Lisp, Prolog ou Squeak sont particulièrement adaptés à la récursivité. Ils agissent sur des listes, c'est-à-dire des mots en langage courant, et sont particulièrement appropriés à certains problèmes. Mais nous avons vu qu'avec un langage conventionnel comme le C nous avons pu traiter des problèmes classiques, quitte à utiliser une programmation hybride, avec des fonctions récursives et en même temps l'usage de tableaux. Nous avons vu cela lors de la programmation du tri par fusion, où les variables de la fonction récursive étaient les indices d'un tableau mis en global. Nous allons maintenant pousser à bout cette technique, en jouant à la fois sur les variables locales de la fonction récursive, et sur des variables globales.

C'est l'occasion de prendre un exemple difficile au premier abord, ce que l'on appelle l'opération de mélange, que nous notons *U*, agissant sur des mots.

L'opération de mélange

Cette opération qui joue sur deux mots est définie comme suit :[1]

- Si un mot *m* a pour longueur 0 (il s'agit du mot vide noté e) alors son mélange avec un mot *m*' est tel que

e U *m*'= *m*' U *e* = *m*'.

- Si les deux mots ont une longueur au moins égale à 1, ils s'écrivent *am* et *bm*' en faisant ressortir leur première lettre, notamment le premier mot *am* signifie que sa première lettre est *a*, et qu'elle est concaténée au reste *m* du mot, alors leur mélange est tel que

am U *bm*' = *a*(*m* U *bm*') + *b*(*am* U *m*'), où le signe + signifie « ou ».

On a là une définition récursive, où l'opération du mélange joue sur des mots de plus en plus courts, avec finalement le test d'arrêt lorsque l'un des mots est vide. Remarquons que l'on n'a jamais les deux mots vides en même temps.

Voyons ce que cela donne sur des exemples :

- Mélange des deux mots 0 et 1 de longueur 1 tous les deux :

0 U 1 = 0(*e* U 1) + 1(0 U *e*) = 01 + 10.

- Mélange de deux mots, l'un de longueur 1 l'autre de longueur 2, soit les mots 0 et 12 :

0 U 12 = 0(*e* U 12) + 1(0 U 2)
= 012 + 1(0(*e* U 2) + 2(0 U *e*))
= 012 + 102 + 120

[1] Pour plus de détails voir les cours de C. Lenormand, notamment le volume *arbres et permutations*, Université Paris 8, disponible sur Internet.

- Mélange de deux mots de longueur 2, soit 01 et 23.

01 U 23 = 0(1 U 23) + 2(01 U 3)
= 0(1(*e* U 23) + 2(1 U 3)) + 2(0(1 U 3) + 3(01 U *e*))
= 0 (123 + 2(1(*e* U 3) + 3(1 U *e*)) + 2(0(1(e U 3) + 3(1 U *e*)) + 301)
= 0(123 + 213 + 231) + 2(013 + 031 + 301)
= 0123 + 0213 + 0231 + 2013 + 2031 + 2301

Ce mélange aboutit à six mots.

Remarques

1) L'opération de mélange est commutative. Mélanger *m* et *m*' revient à mélanger *m*' et *m*. Cela se démontre par récurrence, c'est vrai lorsqu'un mot est vide, et cela se propage lorsque l'on fait grossir les mots d'une unité.

2) Que signifie concrètement le mélange ? Lorsque l'on prend deux mots, l'un qui s'écrit 0 1 2 3 … avec pour longueur *L* et l'autre qui s'écrit 0' 1' 2' 3' … de longueur *L*', leur mélange est formé de tous les mots de longueur *L* + *L*' où les lettres 0 1 2 … apparaissent dans cet ordre ainsi que les lettres 0' 1' 2' … Là aussi la démonstration se fait facilement par récurrence. Autrement dit on a affaire à des permutations spéciales.

3) Les mots fabriqués par le processus récursif sont dans l'ordre alphabétique.

Nombre de mots obtenus lors d'un mélange

Prenons le mélange de deux mots 0 1 2… et 0' 1' 2'… de longueur *L* et *L*', et appelons $m(L, L')$ le nombre de mots du mélange. Grâce à la remarque 2 précédente, ces mots commencent soit par 0 soit par 0'. S'ils commencent par 0, les restes des mots obtenus sont au nombre de $m(L - 1, L')$, et s'ils commencent par 0' les restes des mots sont au nombre de $m(L, L' - 1)$. D'où la relation de récurrence

$m(L, L') = m(L - 1, L') + m(L, L' - 1)$,
avec en conditions initiales : $m(L, 0) = m(0, L) = 1$

Mais on sait, grâce à la formule du triangle de Pascal, que
$C_{L+L'}^{L} = C_{L+L'-1}^{L} + C_{L+L-1}^{L'}$, avec $C_L^0 = 1$. C'est la même suite que la précédente, d'où

$m(L, L') = C_{L+L'}^{L}$ (ou $C_{L+L'}^{L'}$).

Les premières valeurs des nombres $m(L, L')$ sont données dans le tableau suivant, où en lecture diagonale on retrouve le triangle de Pascal.

L' \ L	0	1	2	3	4	5
0	1	1	1	1	1	1
1	1	2	3	4	5	6
2	1	3	6	10	15	21
3	1	4	10	20	35	56
4	1	5	15	35	70	126
5	1	6	21	56	126	252

Programmation des mots du mélange

On part de deux mots. L'un s'écrit 0, 1, 2 …, ($L1$ – 1), de longueur $L1$, et il est enregistré dans un tableau $m[L1]$ déclaré en global. L'autre est noté $L1$, $L1$+1, $L1$+2 …, il a pour longueur $L2$ et il est placé dans un tableau $mm[L2]$. Par exemple $m[2]$ est {0 1} et $mm[2]$ est {2 3}. Le programme doit nous lâcher les mots du mélange, soit 0123, 0213, 0231, 2013, 2031, 2301. Ces résultats sont placés à tour de rôle dans un tableau $b[L1 + L2]$ lui aussi déclaré en global.

Voici comment se présente le début du programme, en fait le programme principal qui appelle la fonction de mélange $U(0, 0, 0)$.

```
#define L1 6
#define L2 6
void U(int i, int j, int etage);
int m[L1],mm[L2],b[L1+L2];  /* déclarations en global */
int compteur;   /* pour compter si l'on veut le nombre de mots du mélange */

main()
{ m[0]=0; m[1]=1; m[2]=2; m[3]=3; m[4]=4; m[5]=5;  /* remplissage des tableaux */
  mm[0]=6; mm[1]=7; mm[2]=8; mm[3]=9; mm[4]=10; mm[5]=11;
  U(0,0,0);
}
```

Passons maintenant à la fonction $U(i, j, etage)$. Elle a trois variables : i est l'indice de la case du tableau m à partir duquel on prend la partie finale du mot concernée, j est l'indice de la case du tableau mm à partir duquel on prend le reste du mot mm, et $etage$ est le numéro de l'étage de l'arborescence récursive où l'on se trouve. Au départ on fait $U(0,0,0)$ puisqu'on part du mot complet m (à partir de la case i=0), du mot complet mm (j=0), et de l'étage 0 de l'arborescence récursive.

Par définition du mélange, $U(i, j, etage)$ se rappelle sur $U(i+1, j, etage+1)$ où le premier mot est diminué de sa première lettre, et aussi sur $U(i, j+1, etage+1)$ où c'est le second mot qui est raccourci. Mais dans le premier cas, la première lettre enlevée du mélange doit être concaténée devant lui, pour cela elle est placée dans le tableau b à l'indice qui est l'étage de l'arborescence où l'on se trouve. Et de même dans le second cas. Au fur et à mesure des appels récursifs les mélanges se font sur des mots de plus en plus cours, et ce qui est sorti et mis devant eux est enregistré dans le tableau b.

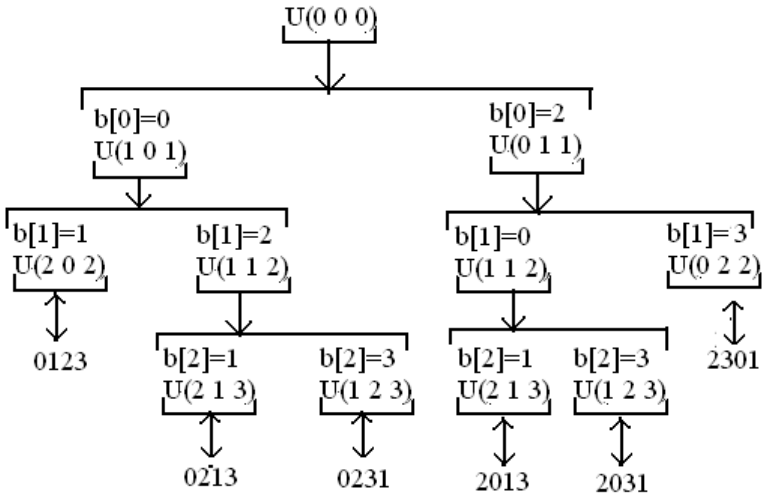
Le test d'arrêt se produit lorsque l'un des mots devient vide, c'est-à-dire pour $i == L1$ ou $j == L2$. On affiche alors le tableau b avec sa partie déjà remplie, et en lui ajoutant le reste de celui des mots qui n'est pas vide.

On en déduit le programme de la fonction U.

```
void U(int i, int j, int etage)
{
int k,q;

if (i==L1)  /* premier mot vide */
  {compteur++;printf("\n%3.d: ",compteur);
   q=0;
   for(k=etage;k<L1+L2;k++) b[k]=mm[j+q++];  /* on rajoute dans b ce qui reste du mot mm */
   for(k=0;k<L1+L2;k++) printf("%d ",b[k]);  /* affichage du mot */
  }
else if (j==L2) /* deuxième mot vide */
  {compteur++; printf("\n%3.d: ",compteur);
   q=0;
   for(k=etage;k<L1+L2;k++) b[k]=m[i+q++];  /* on ajoute le reste du mot m */
   for(k=0;k<L1+L2;k++) printf("%d ",b[k]);  /* affichage du mot */
  }
else if (i<L1 && j<L2)
  {
  b[etage]=m[i];     U(i+1,j,etage+1);  /* on sort la lettre m[i] et on rappelle U */
  b[etage]=mm[j];  U(i,j+1,etage+1);  /* on sort la lettre mm[j] et on rappelle U */
  }
}
```

Mais pour comprendre faisons marcher le programme sur les mots *m* = 01 et *mm* = 23. Autrement dit, dessinons l'arborescence récursive :



C'est seulement là que tout s'éclaire. Car l'arborescence est fabriquée en profondeur. Le premier mot affiché dans le tableau *b* est 0123 en bas de la première branche, celle de gauche. Puis on remonte partiellement, *b*[0] reste à 0, par contre *b*[1] passe à 2 en écrasant l'ancienne valeur 1, et l'on arrive à 0213. Et ainsi de suite. A chaque fois, le tableau *b* est rempli à nouveau partiellement, en conservant son début inchangé. Grâce à sa mise en place en global, il contient à tour de rôle chaque mot en bas des branches successives. Sans le dessin de l'arborescence, on ne serait jamais arrivé à ce programme finalement assez simple.

Retour à l'itératif

Patatras ! Après nous être livré à cette virtuosité récursive, on s'aperçoit qu'il existe une solution beaucoup plus élémentaire. En effet, on a vu que le nombre de mots de mélange, avec L lettres numérotées 0, 1, 2, … et L' lettres notées 0', 1', 2', …, était $m(L, L') = C_{L+L'}^{L}$. Il suffit alors de reprendre le programme des combinaisons, fabriquant tous les mots de longueur $L + L'$ avec L lettres 0 et L' lettres 1. Lors de la lecture de ces mots, il reste à remplacer les 0 par les numéros successifs, 0, 1, 2, … et les 1 par les numéros 0', 1', 2', …, et l'on obtient tous les mots de mélange. Par exemple pour $L = 3$ et $L' = 3$, la combinaison s'écrivant 0 1 1 0 1 0 0 devient le mot 0 0' 1' 1 2' 2 3. D'où le programme :

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define L 5
#define LL 5  /* LL est ce que l'on a appelé L' */
void afficher(void);
int a[L+LL], compteur;

int main()
{
   int i,pospivot,g,d,aux;
   for(i=0;i<L+LL;i++) if (i<L) a[i]=0;else a[i]=1;
   compteur=1;  /* compteur numérote les mots de mélange */
   for(;;)
     {
        afficher();
        i=L+LL-1; while ( i>=1 && !( a[i]==1 && a[i-1]==0)) i--;
        pospivot=i-1; if (pospivot==-1) break;
        a[pospivot]=1; a[pospivot+1]=0;
        g=pospivot+2; d=L+LL-1;
        while(g<d)  { aux=a[g];a[g]=a[d];a[d]=aux;g++;d--;}
        compteur++;
     }
   getch();return 0;
}

void afficher(void)
{   int i,k,kk;
   k=0;kk=0;
   printf("\n%5.d :  ",compteur);
   for(i=0;i<L+LL;i++)    if(a[i]==0) printf("%d  ",k++);   else printf("%d' ",kk++);
}
```