Exercices sur les fonctions

1 Décalage circulaire d'un entier long non signé

Ecrire une fonction qui prend comme paramètre un entier de type `unsigned long` et le transforme en son décalé circulaire.

2 Incrémentation des éléments de la diagonale d'une matrice entière lue dans un fichier

1. Ecrire une fonction qui lit une matrice entière et retourne un tableau à deux dimensions contenant ses coefficients. La matrice sera décrite dans un fichier ayant la forme suivante : la première ligne contient le nombre de lignes de la matrice, la deuxième ligne son nombre de colonnes ; les lignes suivantes correspondent aux coefficients. On utilisera l'opérateur de redirection d'UNIX "<" pour lire le fichier.

2. Ecrire une fonction qui prend comme paramètre une matrice d'entiers (de type `int **`) et qui affiche ses coefficients.

3. Ecrire une fonction qui prend comme paramètre une matrice d'entiers et qui incrémente tous les éléments de sa diagonale.

4. Le nombre de lignes et le nombre de colonnes de la matrice, ainsi que le nom du fichier contenant ses coefficients, sont maintenant passés en arguments du programme.

   - Ecrire une nouvelle fonction de lecture, qui prend comme paramètre une chaîne de caractères (correspondant au nom du fichier contenant la matrice), et qui retourne un tableau d'entiers à deux dimensions stockant les coefficients de la matrice.
   - Modifier le programme principal en conséquence.

3 Distribution des poids d'un code linéaire binaire

3.1 Lecture et stockage de la matrice génératrice

Afin de minimiser à la fois la taille prise en mémoire pour stocker la matrice génératrice et le temps de calcul, on stocke la matrice génératrice binaire dans un tableau à deux dimensions dont chaque ligne est constituée d'entiers de type `unsigned long` (64 bits sur un DEC alpha). De cette façon, on peut diviser la taille de chaque ligne par un facteur proche de `8 * sizeof(unsigned long)`. Par exemple, le vecteur binaire de longueur 10, 1001000101 peut être stocké sur un `unsigned long` dont la valeur (en base 10) est $2^0 + 2^3 + 2^7 + 2^9 = 649$.

On travaillera donc dans toute la suite sur une matrice génératrice compacte de type `unsigned long **`. Pour cela, écrire une fonction qui lit la matrice binaire telle qu'elle est écrite dans le fichier et qui retourne la matrice compacte correspondant. Les longueur et dimension du code, ainsi que le nombre de colonnes de la matrice compacte pourront être déclarées comme des variables globales. On veillera à ce que le programme soit portable, en particulier à ce qu'il fonctionne également sur des architectures où un entier de type `unsigned long` est codé sur 64 bits.

Le prototype de la fonction à écrire sera :

```
unsigned long **lire_matrice(void);
```

3.2 Poids de Hamming d'un vecteur binaire

Ecrire une fonction qui prend comme paramètres un vecteur de type `unsigned long *` et sa longueur, et qui retourne le poids de Hamming de ce vecteur. Son en-tête sera :

```
int poids(unsigned long *vecteur, int taille_vecteur)
```

3.3 Distribution des poids

En se servant de la fonction précédente, écrire un programme qui calcule la distribution des poids du code.

Pour énumérer les 2^k combinaisons linéaires des lignes de la matrice génératrice, on se servira de la méthode suivante utilisant le *code de Gray*. Cet algorithme permet d'énumérer tous les vecteurs de $\mathbf{F}_2^k$ en ne modifiant qu'une seule coordonnée à chaque étape. Les vecteurs de $\mathbf{F}_2^k$ sont représentés par k bits, $(g_0, \dots, g_{k-1})$. On ajoute à ce vecteur une composante g_k vérifiant à tout moment $g_k = \sum_{i=0}^{k-1} g_i \bmod 2$.

INITIALISATION :

$$g_0 = g_1 = \dots = g_{k-1} = g_k = 0 \ .$$

A CHAQUE ÉTAPE :

- Si $g_k = 0$, modifier g_0.
- Sinon :
  - $i \leftarrow$ le premier indice tel que $g_i \neq 0$.
  - si $i \neq k-1$, modifier g_{i+1}.
  - si $i = k-1$, modifier g_{k-1}.

Remettre à jour g_k.

Ainsi, les vecteurs de $\mathbf{F}_2^2$ seront énumérés dans l'ordre suivant : (0,0) - (1,0) - (1,1) - (0,1)

On peut utiliser cette méthode pour énumérer toutes les combinaisons linéaires de k lignes. L'indice de la coordonnée à modifier à chaque étape de l'algorithme précédent correspond à l'indice de la ligne à ajouter à la combinaison linéaire précédente. Par exemple, pour $k = 2$, on énumérera les combinaisons linéaires dans l'ordre suivant : vecteur nul, ligne 1, ligne 1 + ligne 2 , ligne 2.

3.4 Passage des caractéristiques du code en arguments du programme

Modifier le programme précédent afin que la longueur, la dimension et le nom du fichier où la matrice génératrice est stockée figurent en arguments du programme. Pour appeler le programme, on tapera par exemple :

```
a.out 10 3 fichier
```