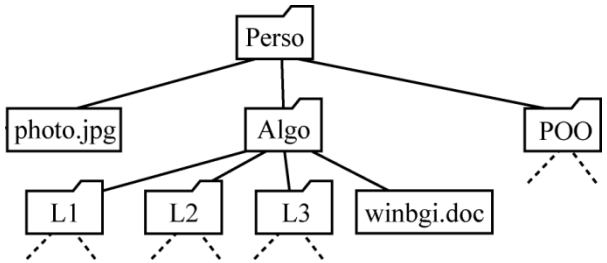
Arbres n-aires appliqués au parcours de répertoires

Dans un arbre n-aire, chaque nœud peut avoir un nombre quelconque de fils. Les répertoires dans un système de fichiers sont généralement organisés sous forme d'arbre n-aire.

Exemple



Les fonctions qui permettent d'ouvrir des répertoires et de lire leurs contenus dans un programme C++ sont proches de celles qui permettent d'ouvrir et de lire des fichiers en C.

Exemple :

```
#include <dirent.h>

DIR *rep;                    // déclaration d'un pointeur de répertoire

rep = opendir("Perso");      // ouverture du répertoire de nom Perso
```

opendir prend en paramètre une chaîne de caractères « à la C » (tableau de caractères). Le pointeur rep est positionné au début du répertoire Perso. Le pointeur est nul si le répertoire n'a pas pu être ouvert.

```
dirent *elem;                // déclaration d'un pointeur vers un élément de répertoire

elem = readdir(rep);         // lecture du répertoire
```

elem pointe maintenant vers le premier élément du répertoire rep.

```
cout << elem->d_name << endl;     // affiche le nom de l'élément
```

La structure dirent contient différents champs. Le champ d_name est le nom de l'élément (chaîne de caractères « à la C »).

```
elem = readdir(rep);         // lecture de l'élément suivant dans le répertoire
```

Chaque appel à readdir(rep) passe à l'élément suivant dans le répertoire et renvoie son adresse. Quand la fin du répertoire est atteinte, readdir renvoie un pointeur nul.

```
closedir(rep);       // fermeture du répertoire
```

Exercice 1 :

Ecrire une fonction **void ls(const string &nomRep)** qui affiche le contenu du répertoire nomRep. Appliquée au répertoire Perso, la fonction afficherait :

```
photo.jpg
Algo
POO
```

Aide : nomRep.c_str() renvoie la chaîne de caractères « à la C » correspondant à la string nomRep.

Exercice 2 :

1) Ecrire une fonction **lsr(nomRep)** qui affiche le contenu du répertoire nomRep et de ses sous-répertoires. Appliqué au répertoire Perso, l'affichage devra avoir la forme suivante :

```
photo.jpg
Algo
L1
contenu de L1
L2
contenu de L2
L3
contenu de L3
winbgi.doc
POO
contenu de POO
```

Aide (voir au dos) :

- la structure dirent contient un champ d_type qui indique si l'élément est un répertoire (d_type = DT_DIR), ou un fichier régulier (d_type = DT_REG), ou un lien (d_type = DT_LNK)…
- string(ch) permet de construire une string à partir d'une chaîne de caractères « à la C ».

2) Pour le TP : modifier la fonction de telle sorte que l'affichage ait la forme suivante :

```
photo.jpg
Algo
|     L1
|     |     contenu de L1
|     L2
|     |     contenu de L2
|     L3
|     |     contenu de L3
|     winbgi.doc
POO
|     contenu de POO
```

Exercice 3 :

1) Ecrire une fonction **find(ch, nomRep)** qui affiche la liste des éléments du répertoire nomRep (et de ses sous-répertoires) dont le nom contient la chaîne de caractères ch.

Exemple

find("o", "Perso") affiche

```
Perso\photo.jpg
Perso\Algo
Perso\Algo\winbgi.doc
```

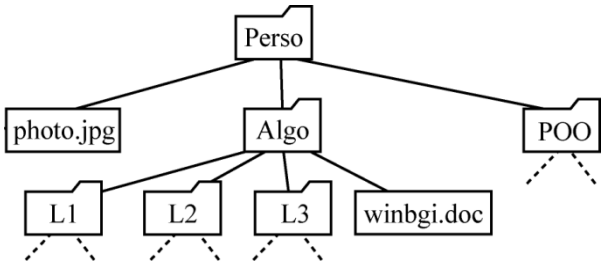
Aide : si ch1 et ch2 sont deux chaînes de caractères de type string, ch1.find(ch2) renvoie l'indice où ch2 commence dans ch1. Si ch2 n'est pas présent dans ch1, la fonction renvoie la constante entière string::npos.

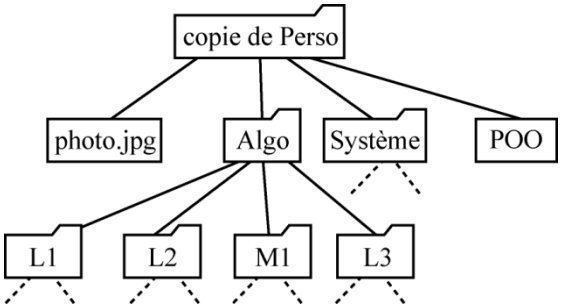
2) Pour le TP : modifier la fonction pour quelle permette d'afficher à la fin le nombre total d'éléments trouvés (3 dans l'exemple précédent).

Exercice 4 :

Écrire une fonction **diff(nomRep1, nomRep2)** qui affiche les différences entre les contenus des répertoires nomRep1 et nomRep2.

Exemple





diff("Perso", "copie de Perso") doit afficher
- Perso\Algo\winbgi.doc
+ copie de Perso\Algo\M1
! Perso\POO
+ copie de Perso\Système

Aide : si rep est un pointeur de répertoire, rewinddir(rep) permet de se repositionner au début du répertoire.