# Extraction de données d'un fichier Excel dans une base de données

## Objectifs du projet

### *Cahier des charges*

Une petite entreprise (PME) de maintenance d'ascenseurs décide d'enregistrer dans une base de données tous ses clients et produits initialement contenus dans des fichiers Excel. Cette opération permettra ensuite à plusieurs utilisateurs (en simultané et même à distance) d'accéder aux données de la PME.

Le logiciel à mettre au point doit extraire d'un fichier Excel toutes les données présentes, les convertir optionnellement au format CSV et XML puis les enregistrer dans une base de données.

On vous fournit 2 fichiers :

- Un fichier au format XLS contenant la liste de tous les clients de l'entreprise : Clients.xls

- Un fichier au format CSV contenant la liste des codes postaux de près de 40000 villes française : CODESPOSTAUX.csv

### *Objectifs pédagogiques*

1. **Ajouter** une **API** (*Application Programming Interface*) dans un projet en C++ et **l'utiliser** en se servant de la documentation et d'exemples fournis.
2. **Créer** un **objet** d'une **classe** et **appeler** les **méthodes** disponibles.
3. **Créer** une **classe** à partir d'un **diagramme de classe** et de la description de ces **méthodes**.
4. **Ouvrir**, **lire** et/ou **écrire** et fermer un **fichier texte** en utilisant la **bibliothèque standard** du C++ (STD).
5. **Mettre en place** un **serveur** de **base de données**. Y **ajouter** une nouvelle **base de données** et des **tables**. **Exécuter** des **requêtes SQL** (*Standard Query Language*).

# Séance 1 (8h)

## *Indicateurs temporels*

| Activités | 1h | 2h | 3h | 4h | 5h | 6h | 7h | 8h |
|---|---|---|---|---|---|---|---|---|
| 1) Ouvrir le fichier Clients.xls et lire le contenu d'une case | | | | | | | | |
| 2) Afficher toutes les données du fichier Clients.xls dans le terminal | | | | | | | | |
| 3) Comparer les codes postaux des clients avec ceux du fichier CODESPOSTAUX.cvs | | | | | | | | |
| 4) Convertir les données du fichiers Clients.xls aux formats CVS et XML | | | | | | | | |
| 5) Afficher sur Google Maps la position d'un client | | | | | | | | |

## *Activité 1 - Ouvrir le fichier Clients.xls et lire le contenu d'une case.*

L'annexe 1 décrit l'API BasicExcel. Cette API vous facilitera grandement la lecture d'un fichier Excel.

Lire le fichier Excel *Clients.xls* et afficher le contenu de la première case de la première feuille du classeur.

## *Activité 2 - Afficher toutes les données du fichier Clients.xls dans le terminal*

Afficher dans une console toutes les données présentes dans ce fichier. Utilisez l'objet cout de la bibliothèque standard.

## *Activité 3 - Comparer les codes postaux des clients avec ceux du fichier CODESPOSTAUX.cvs*

En utilisant la classe ifstream (voir la description de cette classe sur des sites de références comme cplusplus.com, cppreference.com), lire le fichier CODESPOSTAUX.csv et vérifier les codes postaux saisis dans le fichier Clients.xls.

## *Activité 4 - Convertir les données du fichiers Clients.xls aux formats CVS et XML*

En utilisant la classe ofstream, créer le fichier Clients.cvs à partir du fichier Clients.xls. Vérifier que le format cvs est bien respecté.

Proposer ensuite un format XML d'enregistrement des données et créer le fichier Clients.xml correspondant. Vérifier que le format XML est bien respecté.

## *Activité 5 - Afficher sur Google Maps la position d'un client*

# Séance 2

## *Première partie : bilan de la première séance*

Correction de la séance précédente : vous trouverez sur le répertoire de partage un fichier nommé Seance1.cpp contenant la correction.

Testez ce programme. Vérifiez que tous le clients contenus dans le fichier xls sont correctement affichés.

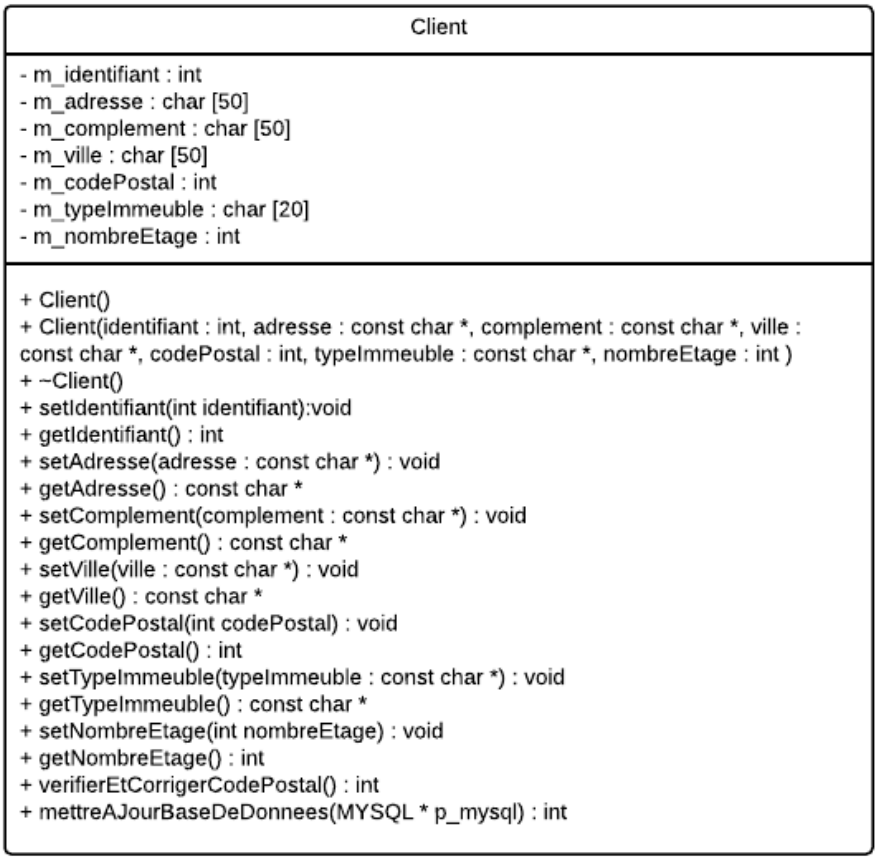
Chaque client est stocké dans une structure nommée Client. Cependant, pour des besoins d'amélioration, nous allons créer une classe Client qui aura les mêmes attributs que la structure précédente mais qui possédera des méthodes qui permettront : a) de vérifier que les code postaux sont corrects et b) de mettre à jour la base de données des clients.

## *Deuxième partie : création de la classe Client.*

### *Mise en place de la classe Client à partir d'un diagramme de classe*

Voici une description UML de la classe Client. Mettez cette classe en place. De l'aide vous sera apportée par la suite pour coder les 2 méthodes suivantes (ne les codez donc pas pour le moment) :

- *verifierEtCorrigerCodePostal()*
- *mettreAJourBaseDeDonnees()*



Modifiez le programme principal pour qu'il complète correctement le tableau contenant tous les clients de l'entreprise en appelant les méthode d'accès appropriées.

### *Vérification du code postal.*

Nous allons maintenant ajouter la méthode *int verifierEtCorrigerCodePostal()* dans la classe Client.

Cette méthode doit :

- Ouvrir le fichier CODESCODESPOSTAUX.csv en utilisant la classe ifstream (voir annexe 1).
- Extraire la première ligne sans l'analyser (voir la description de la méthode *getline()* en annexe 1).
- Pour chaque ligne, extraire le nom de la ville (voir la description de la méthode *getline()* en annexe 1) et le comparer (strcmp) au nom de la ville du client (l'attribut m_ville). S'il sont identiques, il faut alors extraire la donnée suivante : le code postal (voir l'extraction d'un entier dans l'annexe 1). Si les codes postaux sont différents, modifiez l'attribut m_codePostal avec la valeur correcte.

Valeur de retour de la méthode *verifierEtCorrigerCodePostal()* :

La méthode *verifierEtCorrigerCodePostal()* renvoie la valeur

- 1 si le code postal a été modifié ;
- 0 si la ville et le code postal sont déjà corrects ;
- -1 si le fichier CODESPOSTAUX.csv ne s'ouvre pas ;
- -2 si la ville n'a pas été trouvée dans le fichier CODESPOSTAUX.csv.

À partir des informations précédentes et de l'annexe 1 fournie, écrivez la méthode *verifierEtCorrigerCodePostal()*.

Testez cette nouvelle méthode en l'appelant pour chacun des clients extraits du tableau Excel.

## *La base de données*

### *Création de la base de données*

Vous allez maintenant créer la base de données avec le portail phpmyadmin disponible dans la plupart des solutions WAMP.

Pour avoir des solutions uniformes, le nom de la base de données sera "pme". Le nom des tables : "client" et "codepostal". Ajoutez un utilisateur MySQL n'ayant pas les droits d'administration nommé "btsiris" et dont le mot de passe est "btsiris".

### *Le connecteur en C de MySQL*

Les développeurs de la base de données MySQL ont fournit des API permettant de créer facilement des clients MySQL. Ces API sont appelées *connector* (en anglais). Vous en trouverez une dans le répertoire de partage : MySQL.rar.

- Téléchargez cette archive, désarchivez-la à la racine de votre disque dur, soit C:\.
- Copiez le fichier C:\MySQL\lib\libmysql.dll dans votre répertoire de travail : là où il y votre projet C++ Builder...
- Dans C++ Builder, faîtes Projet -> Options puis choisissez l'onglet "Répertoires/Conditions". Ajoutez ";C:\MySQL\include" sur la première ligne "Chemin d'inclusion".
- Maintenant, faîtes Projet -> Ajoutez au projet puis sélectionner le fichier : C:\MySQL\lib\libmysql.lib.

Normalement, tout est mis en place pour pouvoir développer un client MySQL... ce que nous allons nous empresser de faire.

### *Connexion au serveur MySQL*

Dans votre programme principal, vous allez établir une connexion avec le serveur MySQL. Suivez pour cela les instructions données en annexe 2.

### *Ajout de la méthode de mettreAJourBaseDeDonnees() dans la classe Client*

L'argument fourni à cette méthode est un pointeur vers l'identifiant de connexion au serveur MySQL. Cet identifiant est impératif pour faire des opérations sur la base de données.

Dans un premier temps, cette méthode va insérer dans la table 'client' le client courant grâce à la requête SQL INSERT INTO.

Dans un second temps, cette méthode devra :

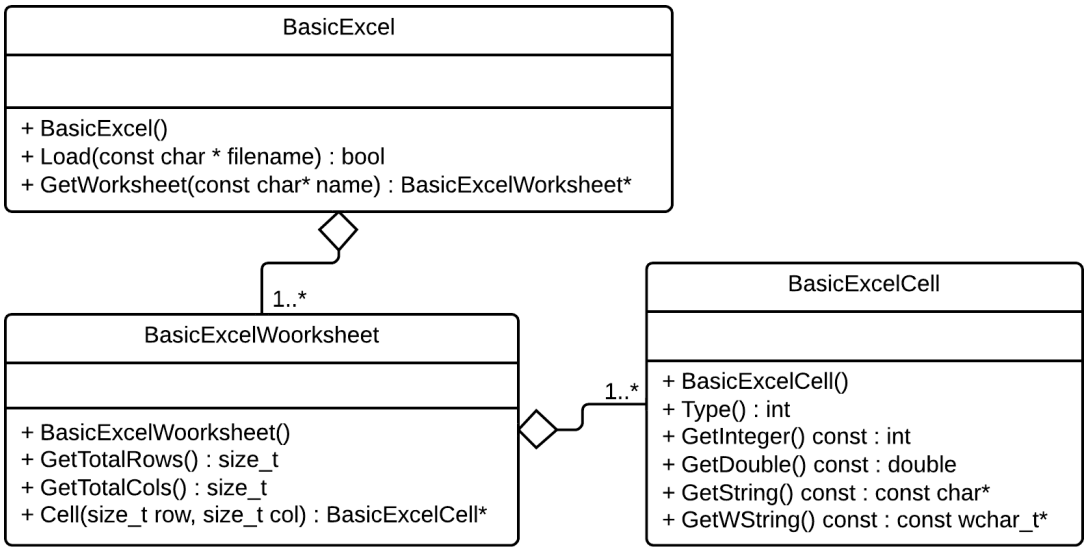
- Insérer le client courant (INSERT INTO) si son numéro d'identifiant n'est pas déjà présent dans la base de données.
- Mettre à jour le client (UPDATE) si son numéro d'identifiant est déjà présent dans la base de données.

# Annexe 1 - L'API BasicExcel

L'API BasicExcel est disponible gratuitement. Elle permet de réaliser des opérations basiques sur des fichiers excel. Elles comblera largement nos besoins exprimés dans le cahier des charges d'introduction.

## Vue d'ensemble

Toute l'API est contenue dans les fichiers open source *BasicExcel.h* et *BasicExcel.cpp*. Ces deux fichiers contiennent la déclaration et la définition de plusieurs classes. Toutes les classes sont définies dans l'espace de nommage YExcel. Nous utiliserons trois classes dont voici un diagramme de classe suivi d'une description sommaire.




La classe **BasicExcel** permet d'ouvrir et de fermer un fichier Excel, couramment appelé un classeur. Elle permet également d'accéder aux feuilles (appelées *worksheet* en anglais) de ce classeur.

La classe **BasicExcelWorksheet** gère une feuille d'un classeur Excel. On peut ainsi connaître le nombre de ligne et de colonne d'une feuille et accéder aux cellules (appelées *Cell* en anglais).

La classe **BasicExcelCell** gère une cellule d'une feuille. On peut facilement connaître le type et le contenu de la cellule.

Avant de rentrer dans le détail de ces classes, expliquons comment ajouter l'API BasicExcel dans un projet.

## Ajout de l'API au projet

Copier les fichiers *BasicExcel.h* et *BasicExcel.cpp* dans votre répertoire de travail. Ajoutez ensuite le fichier BasicExcel à votre projet (Projet -> Ajouter au projet).

Il vous est maintenant possible d'inclure le fichier d'entête de l'API :

```
#include "BasicExcel.h"
```

Et puisque toutes les classes sont définies dans un espace de nommage (YExcel), ajoutez :

```
using namespace YExcel;
```

## La classe BasicExcel

- d'ouvrir un fichier Excel (format XLS) avec la méthode bool Load(const char * filename).

- d'extraire une feuille d'un fichier Excel par son nom avec la méthode BasicExcelWorksheet* GetWorksheet(const char* name). Cette méthode renvoie un pointeur vers un objet de type BasicExcelWorksheet.

## La classe BasicExcelWorksheet

- de connaître le nombre de ligne de la feuille avec la méthode size_t GetTotalRows().

- de connaître le nombre de colonne de la feuille avec la méthode size_t GetTotalCols().

- de récupérer un pointeur vers une cellule par la méthode BasicExcelCell* Cell(size_t row, size_t col). Cette méthode renvoie un pointeur vers un objet de type BasicExcelCell.

## La classe BasicExcelCell

- de connaître le type d'une cellule avec la méthode int Type(). Cette méthode renvoie une valeur définie par l'énumération suivante : enum {UNDEFINED, INT, DOUBLE, STRING, WSTRING};

- d'extraire les données présentes dans une cellule avec les méthodes suivantes :

- int GetInteger() const : extraction d'un entier

- double GetDouble() const : extraction d'un réel

- const char* GetString() const : extraction d'une chaîne de caractères (ANSI String)

- const wchar_t* GetWString() const : extraction d'une chaîne de caractère (Unicode String)

# Annexe 2 - La manipulation des fichiers en C++

## Les classes de la bibliothèque standard du C++

Dans la librairie standard du C++, il existe 3 classes de manipulation de fichiers :

- ifstream pour la lecture dans un fichier (la documentation)
- ofstream pour l'écriture dans un fichier (la documentation)
- fstream pour la lecture/écriture dans un fichier (la documentation).

La classe qui nous intéressera dans ce TP est la classe ifstream puisque nous allons lire un fichier d'extension csv.

## Ouverture et fermeture d'un fichier

Après avoir créé un objet de type ifstream avec le constructeur sans argument, il faut appeler la méthode *open()* de cette classe pour ouvrir un flux en lecture vers un fichier. Vous trouverez ici la documentation de la méthode *open()* ainsi qu'un exemple.

La fermeture d'un fichier se fait par l'appel de la méthode *close()* dont la documentation est ici. Un fichier doit toujours être fermé pour éviter des problèmes d'accès.

## Lecture du fichier

### *Extraction d'une ligne entière*

L'extraction d'une ligne entière se fait grâce à la méthode *getline()* de la classe ifstream. Cette méthode extrait donc toutes les données du fichier jusqu'à rencontrer le caractère de fin de ligne '\n'. Voici la documentation ici. Voici un exemple d'utilisation de cette méthode :

```
ifstream fichier;
char ligne[100];
fichier.open("toto.txt");
fichier.getline(ligne, 100);
cout << "Premiere ville du fichier : " << ligne << endl;
```

### *Extraction jusqu'à un délimiteur*

Cette méthode *getline()* existe aussi avec 3 arguments. Elle permet alors de spécifier le délimiteur d'extraction. Dans l'exemple précédent, le délimiteur d'extraction était le caractère '\n', mais pour des raisons spécifiques, on peut avoir besoin d'extraire des données jusqu'à un ';' par exemple. Le code deviendrait le suivant :

```
ifstream fichier;
char ville[50];
fichier.open("toto.txt");
fichier.getline(ville, 50, ';');
cout << "Premiere ville du fichier : " << ville << endl;
```

### *Extraction d'un entier, d'un mot, d'un caractère...*

D'un façon générale, l'extraction se fait avec l'opérateur de lecture sur un flux >>. Pour extraire un entier, un mot puis un caractère tous séparés par des espaces, on écrirai :

```
int entier;
string mot;
char caractere;
fichier >> entier >> mot >> caractere;
```

### *Autres méthodes utiles : peek(), eof(), fail()*

La classe ifstream possède de nombreuses méthodes dont voici quelques-unes pouvant vous être utiles :
*fail()* : Indique qu'une erreur est survenue par exemple lors de l'ouverture du fichier, lors de la lecture à un endroit inapproprié, etc. La documentation de cette méthode est ici.
*eof()* : Indique que la fin du fichier est atteinte. La documentation de cette méthode est ici.
*peek()* : lit le caractère suivant sans l'extraire. La documentation de cette méthode est ici.

# Annexe 2 - L'API C MySQL

## Inclusion et déclaration

Pour utiliser l'API C de MySQL, il convient d'inclure le fichier d'entête suivant :

```
#include <mysql.h>
```

De plus, il est nécessaire de posséder un identifiant de connexion MySQL. La déclaration se fait ainsi :

```
MYSQL mysql;
```

Ensuite, il faut initialiser l'objet précédemment créé avec la fonction *mysql_init()* dont une description est disponible ici. On pourra alors écrire :

```
mysql_init(&mysql);
```

## Connexion à une base de données

La connexion à une base de données s'effectue par la fonction *mysql_real_connect()* disponible dans cette API. Vous trouverez une documentation et un exemple d'utilisation de cette fonction ici.

## Émission d'une requête

Une fois la connexion établie, il est possible d'envoyer une requête SQL à votre serveur. Les requêtes SQL peuvent être INSERT, DELETE, UPDATE, SELECT, etc. Vous trouverez une documentation claire sur les requêtes SQL avec de nombreux exemples ici.

La fonction permettant d'envoyer une requête au serveur MySQL est *mysql_query()*. Vous trouverez une documentation complète de cette fonction ici.

Le site MySQL est un site de référence qu'il est judicieux de consulter pour obtenir la documentation de l'API en C de MySQL ainsi que sur la syntaxe SQL.