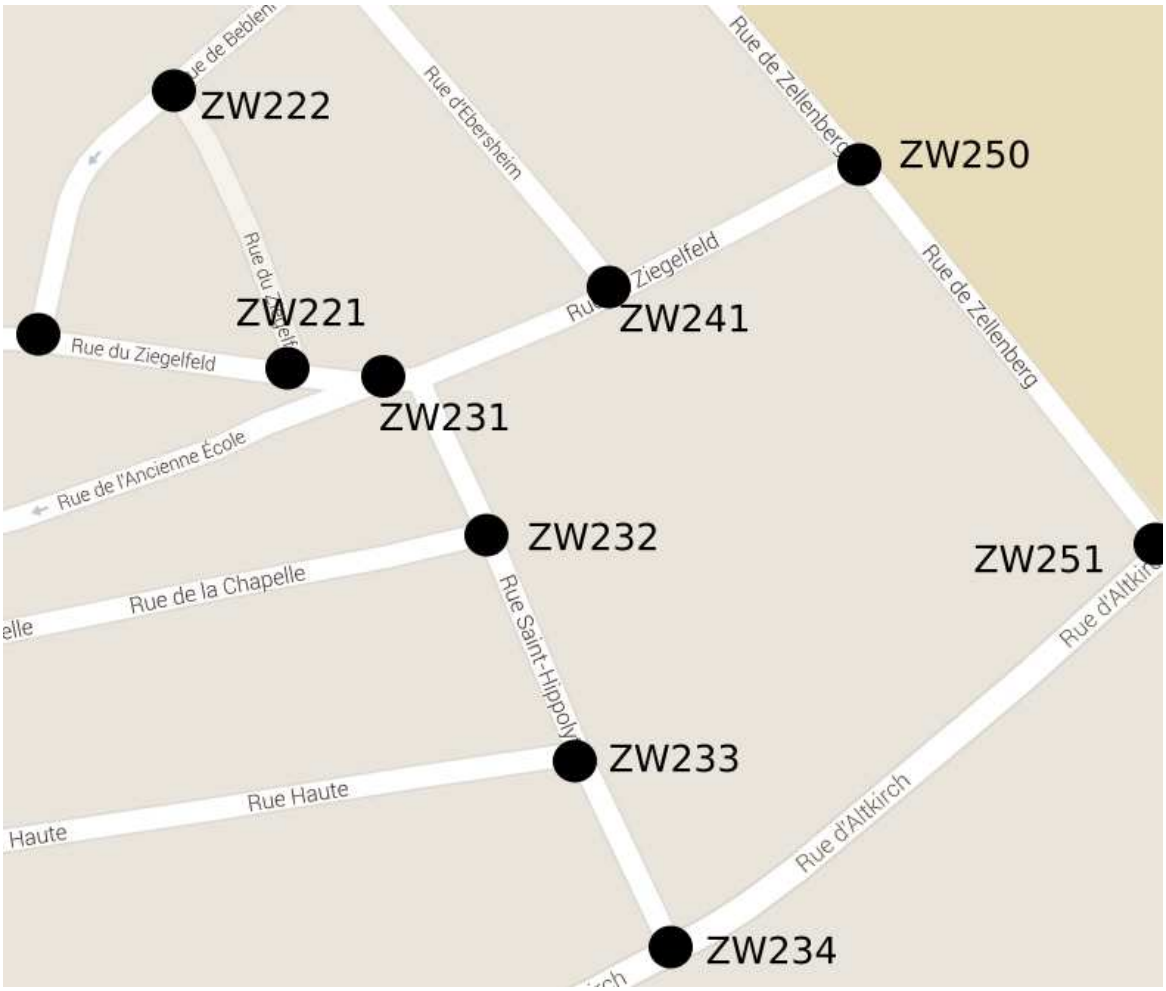
Travaux Pratiques
séance nr. 3 - Graphe de voierie urbaine

Dans ce TP nous allons implanter une structure de "graphe orienté".

On souhaite modéliser un réseau de voierie urbaine recensant voies publiques et carrefours. Pour cela on construit un graphe valué dont les sommets sont les carrefours et les arêtes les rues. Chaque carrefour possède un nom qui est un code : 2 lettres suivies de 3 chiffres (un nombre). Chaque rue possède un nom (une chaîne de caractères de longueur arbitraire). Certaines rues sont à sens unique : c'est pourquoi le graphe est orienté.

Le schéma ci-dessous présente un exemple. Les points noirs sont les sommets qui marquent les carrefours. Chaque carrefour a un nom (ici, ZW231, ZW232, etc.). Chaque rue possède également un nom. Une même rue peut comprendre plusieurs carrefours, donc correspondre à plusieurs arêtes dans le graphe : voir la « rue Saint-Hippolyte ». Les flèches indiquent le sens unique, par exemple « rue de l'ancienne école ».



Les rues sont stockées à l'aide d'une **table de chaînes de caractères**, et les carrefours à l'aide d'une seconde table de chaînes de caractères (reprendre la structure *table* vue au TP2). Ces deux tables sont de même type, mais séparées : il y donc deux tables Chaque carrefour et chaque rue est identifiée par un entier non signé. Nous modélisons la structure ***GrapheUrbain*** en utilisant une **liste d'adjacence**. La structure est la suivante :

```
typedef struct strarc {unsigned int icf, irue; struct strarc *suc;} Strarc, *ListeSom;

typedef struct {unsigned int icf; ListeSom lsucc, lpred; }  Sommet;

typedef struct strsom {Sommet  c; struct strsom *suiv;} Strsom, *GrapheUrbain;
```

La première table est utilisée pour associer l'identifiant du carrefour ***unsigned int icf*** à la chaine de caractères (nom du carrefour). Le seconde table est utilisée pour associer l'identifiant de la rue ***unsigned int irue*** à la chaine de caractères (nom de la rue).

Notez que cette structure diffère de celle vue au TD4 : ici nous modélisons à la fois la liste des successeurs (*lsucc*) et la liste des prédécesseurs (*lpred*) pour chaque sommet. Par ailleurs le graphe est valué : chaque arc porte le nom d'une rue.

Pour chacune des opérations suivantes proposer un profil et indiquer les préconditions.

3.1 Définir et implémenter les opérations de création et destruction d'un ***GrapheUrbain***.

3.2 Définir et implémenter des opérations d'adjonction / suppressions de rue et de carrefour.

3.3 Implémenter les opérations usuelles sur le graphe et notamment une opération permettant d'ajouter une arête au graphe, donc de relier deux carrefours par une rue. Les autres opérations sont : l'existence d'une rue, l'existence d'un carrefour, l'existence d'une connexion entre deux carrefours (càd qu'un carrefour fait partie des successeurs/prédécesseurs d'un autre carrefour).

Ecrire un code « main » qui permet de construire le graphe donné à titre d'exemple.

3.4 Implémenter une opération permettant, pour un carrefour donné, d'obtenir la liste de toutes les rues incidentes. Ajouter dans le main un code qui teste cette opération. Par exemple, l'utilisateur saisit: « ZW233 » et on affiche : rue Haute, rue Saint Hippolyte.

3.5 Implanter une opération permettant, pour une rue donnée (on fournit en paramètre une chaîne de caractères), de récupérer la liste de toutes les rues qui la croisent. Ecrire dans le main un test. Par exemple, pour la « rue Zellenberg » on affiche : rue Ziegelfeld, rue d'Altkirch.