



# Mini-projet 1 en Programmation C

## Simulation d'un réseau routier de bus

**Version Release 1 : 05/02/2024**

Département : Technologies de l'informatique	A.U : 2023 – 2024 – semestre1
Niveau : 1ère année Tronc Commun Eléments Constitutifs : Algo&Str de données, Programmation C	Enseignant: amine tarhouni

### 1. Objectif

L'objectif du mini-projet est d'implémenter un programme en C visant à simuler (de la façon la plus proche possible de la réalité) un réseau routier de bus dans une ville.

### 2. Description Générale

Une ville est composée de plusieurs zones accueillant des personnes, qui peuvent être des habitants de cette zone, des travailleurs ou des clients qui font leurs courses dans la zone.

Une personne habitant dans une zone donnée, chaque jour, se déplace d'une zone à une autre par bus en payant des tickets, pour effectuer un ensemble d'activités (se rendre à son travail s'il en a, faire ses courses ou se promène dans un centre commercial...), la journée se termine toujours par le retour de la personne à son lieu d'habitat.

Les zones sont reliées par des routes parcourues par des bus transportant des passagers, chaque bus est affecté à une ou plusieurs lignes, qu'il peut parcourir plusieurs fois par jour dans les deux sens.

Une ligne de bus est composée de plusieurs tronçons de route, chaque tronçon de route unitaire est borné par deux zones successives. Un tronçon entre deux zones successives comporte à son tour plusieurs subdivisions unitaires (correspondant à 100 mètres par exemple).

### 3. Description De données

#### Les enregistrements

- Un enregistrement **horaire** possède les champs suivants :

```
typedef struct {  
    int minute;  
    int heure;  
    int jour;  
} Horaire ;
```

- Un enregistrement **zone** possède les champs suivants :

```
typedef struct {  
    int id;  
    int row;  
    int column;  
    int nombrePersonnes;  
    Personne personnes[100];  
} Zone ;
```

Une zone se situe à la position de coordonnées (row, column) dans la carte géographique, et comporte un tableau de personnes qui sont présents dans cette zone.

- Un enregistrement **tronçon** possède les champs suivants :

```
typedef struct {  
    int idZoneSrc;  
    int idZoneDest;  
    int distance;  
    Horaire dureeApproximative;  
    int nombresubdivisions;  
    Subdivision parcours[20];  
} Troncon ;
```

Un tronçon s'étale d'une zone src à une zone dest successives, qui a une distance, une durée de parcours et un ensemble de subdivisions unitaire ou tranches de rues unitaires qui le composent.

- Un enregistrement **subdivision** possède les champs suivants :

```
typedef struct {  
    int row;  
    int column;  
    int codeRue  
} Subdivision ;
```

Une subdivision unitaire ou tranche de rue unitaire possède des coordonnées (row, column) dans la carte géographique. Le champ codeRue indique si cette tranche de rue est horizontale, verticale, ou autre (les codeRue vont être détaillés dans ce qui suit).

- Un enregistrement **personne** possède les champs suivants :

```
typedef struct {  
    int cin;  
    int idZoneHabitat;  
    int idZoneTravail;  
    int idZoneCourante;  
    int intentionVoyage;  
    int idLigneCourante;  
    int sensCourant;  
    int idZoneDest;  
    int nombreActivites;  
    Activite activites[8];  
} Personne ;
```

Une personne possède un cin unique, une idZoneHabitat et idZoneTravail qui restent inchangés, une idZoneCourante de la zone où il se trouve maintenant. Une intentionVoyage dont la valeur 0 signifie que la personne est occupée et ne compte pas voyager, dans le cas où elle veut voyager l'intentionVoyage est mis à 1, et dans ce cas idLigneCourante,

sensCourant et idZoneDest indiquent respectivement la ligne du bus qu'il veut prendre, le sens aller ou retour (0 ou 1), et la zone destination où il veut descendre, enfin un tableau des différentes activités d'une personne et leur nombre.

- Un enregistrement **activité** possède les champs suivants :

```
typedef struct {  
    int id;  
  
    horaire horaireDateSouhaite;  
  
    horaire dureeHeureMinute;  
  
    int typeActivite;  
  
    int idZoneSrc;  
    int idZoneDest;  
    int idLigne;  
    int sens;  
} Activite ;
```

Chaque activité a un horaire souhaité de sortie de la zone src pour rejoindre la zone dest de l'activité qui peut être à domicile, au travail ou commerciale (typeActivite respectivement 0, 1 ou 2), enfin le idLigne et sens représentent la ligne de bus à prendre pour arriver à destination et le sens comme expliqué précédemment .

- Une **ligne** possède les champs suivants :

```
typedef struct {  
    int id;  
  
    int nombreZones;  
  
    int idZones[10];  
} Ligne ;
```

Une ligne comporte une succession des id Zones qui constituent sa trajectoire.

- Un enregistrement **bus** possède les champs suivants :

```
typedef struct {  
    int id;  
    int idZoneCourante;  
    int row;  
    int column;  
    int enRoute;  
    int idLigneCourante;  
    int sensCourant;  
    int nombreTaches;  
    Tache taches[10];  
    int nombrePassagers;  
    Personne passagers[20];  
    int nombreTickets;  
    Ticket tickets[200];  
} Bus ;
```

Un bus se trouve dans les coordonnées (row, column), il a l'id zone courante qui désigne la zone où il est garé ou qu'il vient de dépasser tant qu'il n'est pas arrivé à une autre zone. enRoute est mise à 1 si le bus est en fonctionnement, 0 s'il est en arrêt. Dans le cas où le bus est en route, il suit une ligne courante et un sens courant. L'enregistrement bus stocke aussi ses tâches et leur nombre, ainsi que des copies des tickets vendus et leur nombre.

- Un enregistrement **tâche** possède les champs suivants :

```
typedef struct {  
    int id;  
    int idLigne;  
    int sens;  
    Horaire horaireDateDepart;  
} Tache ;
```

La tâche d'un bus est un voyage suivant une ligne donnée dans un sens donnée (0 pour dire aller et 1 pour dire retour), à un horaire de départ donné.

- Un enregistrement **ticket** possède les champs suivants :

```
typedef struct {  
    int zoneSrc;  
    int zoneDest;  
    int idLigne;  
    int sens;  
    int idBus;  
    horaire dateVente;  
    int distance;  
    int prix;  
} Ticket ;
```

Dans un ticket de bus, on mentionne les zones source et destination, la ligne et le sens suivis, l'id de bus pris, l'horaire de vente, la distance qu'on va parcourir et le prix du ticket.

- Un **caseCarte** possède les champs suivants :

```
typedef struct {  
    int rue;  
    int zone;  
    int bus;
```

Une caseCarte se trouve dans des coordonnées (row,column) bien précis du tableau carte géographique, cette caseCarte contient l'id de la zone qui se trouve dans ces mêmes coordonnées (row,column), sinon la valeur est -1. De même la caseCarte contient l'id du bus et le code la tranche de rue s'il en existe dans ces mêmes coordonnées (row,column), en cas de non existence les valeurs respectivement sont -1 et 0.

## Les Tableaux

- Un tableau **carteGéographique** est un tableau deux dimensions (20\*20) est utilisée pour schématiser la ville et se compose de casesCartes dans dans des coordonnées (row,column) bien déterminés.
- Un tableau **zones** est un tableau de "zone" (10 zones).
- **flotteBus** est un tableau de "bus" (10 bus) : on pourra faire sortir 5 bus de la zone initiale dans le sens "aller" et 5 autres depuis la zone finale de la ligne dans le sens "retour" au rythme d'un bus toutes les 20 minutes.
- **lignes** est tableau de "ligne" de bus (on peut se contenter d'une seule ligne à la première partie, dont on pourra choisir la zone 0 comme zone de départ).
- **troncons** est un tableau composé de "troncon" (10 tronçons) entre deux zones successives décrits précédemment.
- **tickets** est un tableau de "ticket" contient les copies de tickets des différents bus récupérés à la fin de chaque journée (10000 tickets).

## 4. Description De Fonctionnement

### Préparer les fichiers de données :

Avant de lancer le jeu, le programmeur doit préparer des fichiers textes de données correspondant chacun à un des thèmes suivants :

- les zones,
- les bus et leurs taches,
- les tronçons et leurs subdivisions,
- Les lignes

Par exemple, les zones et leurs informations à raison de une zone par ligne, les champs séparés par un symbole ou espace.

Une ligne du fichier qui concerne un bus, pourrait contenir le nombre de tâches comme dernier champ, indiquant le nombre de lignes parmi les lignes suivants du fichier qui comportent les informations des tâches de ce bus en cours, ensuite la ligne du fichier suivant ces tâches va concerner un nouvel autre bus.

La même procédure de remplissage de fichier s'applique aux fichiers tronçons et lignes de bus.

### **Charger des données depuis des fichiers :**

Au début du jeu, ces fichiers seront chargés pour initialiser les tableaux de zones, de bus, de tronçons et de lignes.

Ensuite on initialise, puis on utilise les tableaux précédemment cités pour mettre à jour la carte géographique. Effectivement, la carte géographique qui n'est autre qu'un tableau deux dimensions de caseCarte et dont les champs rue, zone et bus sont qui déjà initialisés comme champs de case vide, vont ainsi se mettre à jour par le numéro de bus et/ou de zone et/ou le code de traçage de la subdivision, dont les coordonnées correspondent aux coordonnées d'une caseCarte donnée. Les codes de traçage de la rue vont être détaillés dans ce qui suit.

### **Générer aléatoirement des personnes :**

Les personnes seront générés aléatoirement (et non pas par chargement depuis des fichiers) notamment le cin qui prendra une valeur unique croissante, les autres champs peuvent être initialisés à -1 ou 0 selon le cas.

Leurs zones, d'habitats et de travail et leurs activités seront générées aléatoirement, sauf que la première activité de chaque journée a toujours comme départ la zone d'habitat (à une heure qui peut être aléatoire entre 6h, 7h ou 8h), et peut avoir comme destination la zone de travail ou une zone aléatoire pour faire des courses.

L'activité suivante fait le départ de la dernière zone où on était, à une zone de travail ou une autre zone aléatoire pour faire des courses, et ainsi de suite jusqu'à la dernière activité d'une journée donnée, qui doit obligatoirement avoir la zone d'habitat de la personne en cours comme zone destination.

Et ces mêmes règles s'appliquent à tous les jours où on a des activités.

Une fois qu'une personne est complètement initialisée, elle sera affectée au tableau de personnes de sa zone d'habitat.

### **Démarrage d'une journée :**

A chaque minute, plusieurs mises à jour seront effectuées.

En effet, on met à jour toute les minutes les informations des personnes (dans les différentes zones) qui manifestent des intentions de voyages et ceci d'après les horaires prédéfinis des activités de chacun d'eux.

On met en état en "enRoute" chaque bus dont l'horaire de départ dans une ligne donnée est atteint.



Chaque bus qui est déjà en route et qui a atteint une zone donnée va débarquer les passager qui ont cette zone comme destination, et ceci en déplaçant les personnes en question depuis le tableau de passagers du bus vers le tableau des personnes existants dans cette zone.

Après une mise à jour des informations des personnes débarquées, on vérifie s'il existe des personnes qui veulent voyager et qui demandent la même ligne et le même sens que le bus en cours. Dans ce cas, on va les embarquer, en les déplaçant du tableau de personnes de la zone vers le tableau de passagers du bus.

Enfin, on va mettre à jour la zone destination du bus en la récupérant depuis le tableau des zones parcourues par une ligne, puis récupérer le tronçon qui a comme extrémités la zone en cours du bus et sa zone destination, ensuite déterminer sur quelle subdivision du tronçon le bus se situe et ce par la comparaisons des coordonnées, enfin passer de la subdivision en cours à la subdivision suivante (en prenant ses coordonnées) tout en tenant compte du sens du bus dans la ligne.

En cas d'arrivée du bus à la dernière zone de la ligne en cours, les passagers seront débarqués, et on met à jour le champ enRoute à 0, et comme ça le bus sera immobilisé jusqu'à l'atteinte de l'horaire de la nouvelle tâche.

Une fois tous les bus et zones mises à jour, on effectue une mise à jour de la carte géographique par les nouvelles positions des bus. Et enfin, on trace cette nouvelle carte géographique.

Et ainsi de suite pour chaque minute.

Il est à noter qu'il est recommandé d'utiliser la fonction getch() qui se bloque à chaque fois et se débloque par un appui sur une touche clavier, on pourra choisir la touche '1' pour l'avancement d'une minute, la touche '2' pour l'avancement de dix minutes, '3' pour une heure et '4' pour une journée et 'q' pour quitter...

### **Tracage de la carte géographique :**

La carte géographique est un tableau deux dimensions de caseCarte, dont les champs zone et bus initialisés à -1 et le champ de rue initialisé à 0.

Chaque case doit correspondre à une écriture à l'écran d'un certain nombre de caractères, par exemple 4 caractères.

- Si une caseCarte a les champs zone et bus négatifs et le champ rue est nul, alors les 4 caractères seront une chaîne vide " ".
- Si une caseCarte a le champ zone qui contient un numéro de zone (positif ou nul), la valeur 2 par exemple, alors les 4 caractères à afficher seront " Z2 ".

- Si une caseCarte a le champ zone négatif et le champ bus qui contient un numéro de bus (positif ou nul), la valeur 5 par exemple, alors les 4 caractères à afficher seront "[b5]".
- Si une caseCarte a le champ zone qui contient un numéro de zone (positif ou nul), la valeur 2 par exemple et le champ bus qui contient un numéro de bus (positif ou nul), la valeur 5 par exemple, alors les 4 caractères à afficher seront "Z2b5".
- Si une caseCarte a les champs zone et bus négatifs et le champ rue positif par exemple une valeur qui correspondant à une tranche de rue horizontale, alors les 4 caractères à afficher sont "====".

### **Initialisation et affichage des rues :**

Les rues sont tracées à partir des valeurs des champs de rue dans les caseCarte du tableau carte géographique.

Les champs rue sont initialisés par défaut à 0 qui correspond à une absence de rue, puis lors du chargement des tronçons et leurs subdivisions depuis le fichier correspondant, le champ rue sera mis à jour par le code dans la subdivision qui a le même numéro de row et column que celui de la caseCarte en question.

Comme exemple, on pourra prendre le codage suivant de subdivision de tronçon :

1 → droite

2 → gauche

4 → bas

8 → haut

On peut combiner ces codes unitaires pour obtenir les nouvelles directions suivantes :

3 = (1+2) → rue à deux sens droite et gauche (rue horizontale) =

12 = (4+8) → rue à deux sens bas et haut (rue verticale) ||








9 = (1+8) → rue à sens droite puis haut 丩

10 = (2+8) → rue à sens gauche puis haut ㄥ

11 = (1+2+8) → rue droite haut et gauche haut 丩

15 = (1+2+4+8) → rond pont contenant tous les sens 卄

Sens	Code	Symbole	Code clavier
Droite Gauche	1, 2 ou 3	=	Alt+205
Bas haut	4, 8 ou 12		Alt+186
Droite et bas	5	ㄣ	Alt+187
Gauche bas	6	ㄥ	Alt+201

Droite gauche et bas	7		Alt+203
Droite et haut	9		Alt+188
Gauche et Haut	10		Alt+200
Droite gauche et haut	11		Alt+202
Droite et bas haut	13		Alt+185
Gauche et bas haut	14		Alt+204
Droite gauche et bas haut	15		Alt+206

## 5. Dates importantes

Etape	Date Limite de Dépôt Plateforme
Dépôts hebdomadaires	Dépôt le dimanche de chaque semaine avant 18h dès le 05/02/2023. Une pénalité est attribuée si l'étudiant ne dépose pas la partie demandée à temps.
<b>Dépôt Final</b>	Dimanche <b>12 Mai</b> 2024 avant 18h

- Les explications de l'enseignant constituent une partie entière et officielle du mini projet en plus du cahier de charges. L'absence de l'étudiant (au cours du semestre et **surtout aux premières semaines**) peut **laisser très probablement des lacunes** dans la compréhension du mini projet et des choix ajoutés par l'enseignant-encadreur.
- **A la suite de chaque suivi ou évaluation, des étudiants ayant déposé un travail seront invités à valider leurs travaux. L'absence le jour de la validation est sanctionnée par un zéro même si le travail est bien déposé.**
- Une "réponse à une question" qui n'est pas conforme à l'énoncé et aux (explications, méthodes, choix et démarche) de l'enseignant ne sera pas acceptée.
- **Les dates de dépôts dans la plateforme sont des dates définitives et doivent être respectées, aucun travail ne sera accepté après la fermeture automatique des espaces de dépôts.**

## 6. Etapes du projet

### Conventions et Consignes techniques

- Un module désigne soit une fonction, soit une procédure.
- Ne pas utiliser des variables globales (seulement les constantes globales sont autorisées), ni pointeurs (sauf si ceci sera demandé clairement dans un module donné)
- Les déclarations des fonctions et procédures seront données et doivent être utilisés comme elles le sont.
- Pour modifier un tableau d'une façon permanente dans une fonction ou procédure il suffit de le faire entrer en paramètres de cette fonction ou procédure.
- **Seul un code exécutable va être accepté. Un code de 5 modules exécutables travaillés minutieusement vaut mieux qu'un code 20 modules non exécutables travaillés avec précipitation ou négligence.**

## Modules et travaux demandés

1. Modules d'initialisation, de mise à jour et d'affichage de la carte géographique.
2. Modules de chargement, affichage et sauvegarde de zones, bus, lignes, (tronçons et rues).
3. Module de recherche de ligne et sens pour aller d'une zone source à une zone destination.
4. Module de calcul de la distance entre une zone source et une zone destination.
5. Module d'affectation d'une personne à une zone.
6. Modules de génération aléatoire, affichage et sauvegarde de personnes.
7. Module de déplacement complet d'une personne d'un tableau personnes à autre.
8. Modules de mise à jour informations des personnes d'une zone et de toutes les zones.
9. Modules d'initialisation d'un ticket, vente d'un ticket, déplacement de tickets du bus au tableau de tous les tickets, et affichage d'un tableau de tickets.
10. Module de détermination du numéro de la tâche, si un bus donné va démarrer une tâche à un horaire donné.
11. Module de calcul du prix d'un ticket.
12. Modules d'embarquement et débarquement des passagers d'un bus.
13. Module de déplacement d'un bus.
14. Modules du déroulement d'une semaine minute par minute et de l'affichage simple et double (comme montré dans les annexes).
15. Module qui utilise les autres modules décrits précédemment (et autres modules si nécessaire), pour dérouler l'application.
16. Autres modules que l'étudiant voit nécessaires, (avec confirmation de l'enseignant de la matière).
17. D'autres modules peuvent être ajoutés lors des prochains releases du cahier de charges.

## 7. Evaluation

La note finale sera la note de l'évaluation finale après validation à laquelle seront retranchées les pénalités de non dépôt des objectifs hebdomadaires à temps ou de travail bâclé.

Un dépôt dans le serveur de dépôt <http://miniprojet.maktabati.tn> doit être fait en respectant le planning de la section "Dates importantes" pour assurer le suivi de l'avancement du développement de l'application. En cas de coïncidence de la séance de la matière mini projet ou du jour du dépôt avec un jour férié ou autre, le dépôt sera toujours obligatoire à temps, sachant que l'étudiant peut être invité à une séance de validation dans une autre date.

## 8. Travail à rendre

- Le mini-projet se fait en "**MONOME**", en langage C et **non pas** en C++, **le graphique doit être le plus simple possible** sans utiliser des effets, des introductions ou des messages encombrant et se concentrer sur les fonctionnalités.
- Le logiciel recommandé : Code Blocks (lien de téléchargement complet avec compilateur : <https://bit.ly/3DOi8oE> )
- Les autres logiciels sont autorisés comme devc++ et autre, en sachant que travailler avec des logiciels différents peut présenter des incompatibilités dont l'étudiant sera responsable.
- La **simplicité** du codage et les **choix adéquats** seront pris en considération dans la notation.
- chacun doit remettre avant la date limite de chaque dépôt:
  - Le code source ou les fichiers demandés doivent être chargés dans **le serveur de dépôt <http://miniprojet.maktabati.tn>**, chaque question dans son emplacement réservé, sachant que la plateforme n'accepte le code que si sa compilation réussit.
  - En cas de problème quelconque vous pouvez contacter l'email suivant au moins une heure avant la fermeture du dépôt : **[miniprojet.isetbz@gmail.com](mailto:miniprojet.isetbz@gmail.com)**.

## 9. Règles et consignes

- La présence aux séances de la matière est obligatoire. En fait, toute explication, choix, recommandation ou détail énoncé en classe est considéré comme partie intégrante de ce cahier de charge général.

- L'étudiant ne doit pas se contenter du cours, et ne doit pas aussi attendre à ce qu'il voit chaque détail en séance de cours pour commencer à l'utiliser dans son travail, en effet le projet comporte des parties qui nécessitent un effort personnel de recherche et de réflexion.
- L'étudiant doit déposer son projet exclusivement dans le serveur de dépôt : <http://miniprojet.maktabati.tn> et non pas par email dans aucun cas, dans les délais prédéfinis. Tout manquement à ce délai sera sanctionné par un zéro. Tout problème dans le dépôt doit être déclaré au moins une heure avant la date de fermeture de dépôt sur l'email : [miniprojet.isetbz@gmail.com](mailto:miniprojet.isetbz@gmail.com).
- Les étudiants sont invités à assister dans les séances de la matière à chaque semaine (il est à rappeler qu'à chaque séance l'étudiant est invité à présenter un travail respectable, dans le cas contraire : un travail bâclé sera sanctionné d'une pénalité pouvant aller jusqu'à -0.5 point dans la note finale.
- Pour chaque absence à une séance de la matière mini projet, va être attribuée une pénalité de -0.5 point à la note finale.
- Après chaque dépôt hebdomadaire, une validation du travail déposé est à faire dans la séance de la matière qui suit le dépôt.
- Après le dépôt final, une validation du travail déposé est à prévoir dès le jour qui suit ce dépôt, le planning sera communiqué à temps à partir du jour même du dépôt ou l'un des jours suivants.
- L'absence le jour de la validation d'un dépôt hebdomadaire est sanctionnée par une pénalité même si le travail est bien déposé.
- L'absence le jour de la validation du dépôt final est sanctionnée par un zéro même si le travail est bien déposé.
- Une "réponse à une question" qui n'est pas conforme à l'énoncé et aux (explications, méthodes, choix et démarche) de l'enseignant ne sera pas acceptée.
- Le travail doit être déposé et validé dans les délais fixés à l'avance même si le jour de dépôt ou de validation coïncide avec un jour férié.

- La recherche de codes d'actions élémentaires et de syntaxe du langage est permise telle que (printf, scanf, boucles, déclarer, remplir ou afficher un tableau, lire ou écrire dans un fichier...). Tandis que la copie d'un ou plusieurs blocs de code qui font partie d'une solution du même projet donné ou d'un projet équivalent, depuis internet ou autre est interdit et considéré comme plagiat.
- En cas de changement majeur de l'énoncé ou du planning, les étudiants seront informés dans le tableau d'affichage de l'iset, sur la plateforme uvt et dans le groupe facebook du miniprojet.
- L'enseignant peut être contacté dans le groupe facebook du mini projet, ou bien par email sur [miniprojet.isetbz@gmail.com](mailto:miniprojet.isetbz@gmail.com).
- Tout cas de COPIAGE ou PLAGIAT sera sanctionné par un ZÉRO pour tous les participants à cette opération, et leurs dossiers peuvent être transmis au conseil disciplinaire.



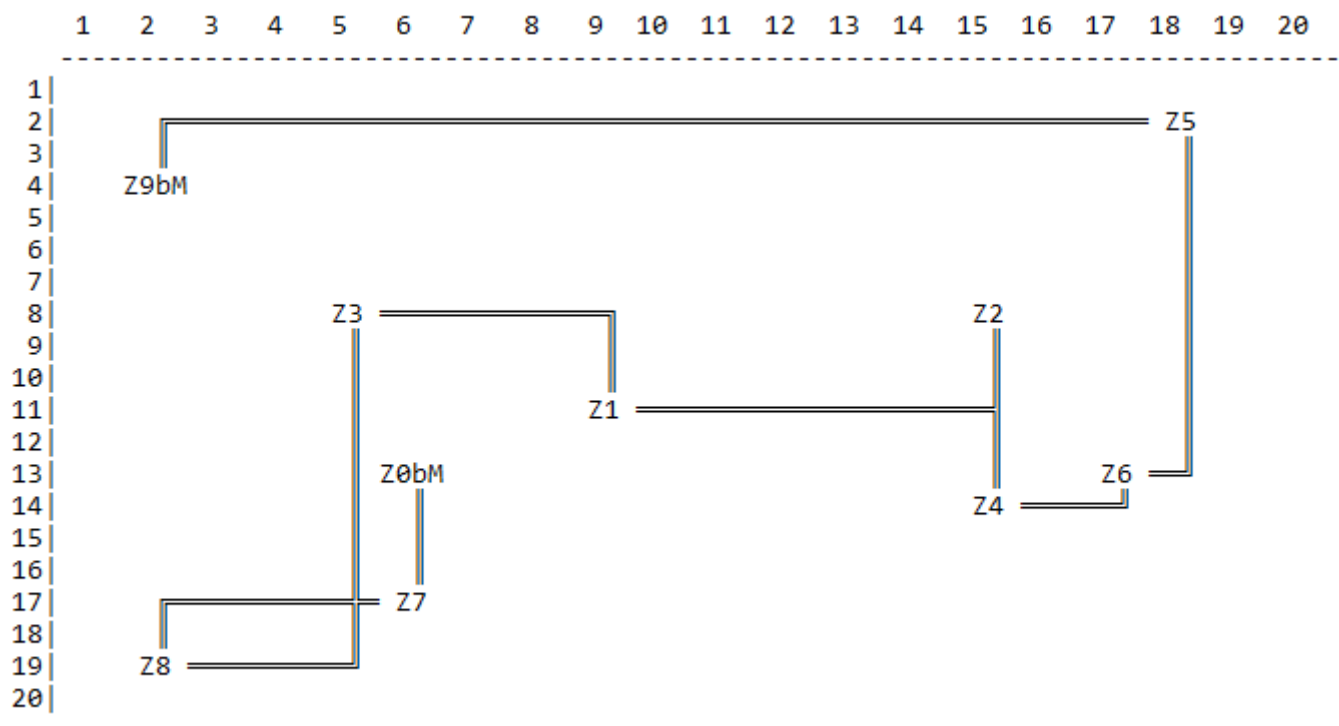
## 9. Annexes

**Annexe (A) :** Carte géographique avant le placement des bus et rues, semblable à la grille de domino du mini projet précédent.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2																		Z5		
3																				
4		Z9																		
5																				
6																				
7																				
8					Z3										Z2					
9																				
10																				
11									Z1											
12																				
13						Z0											Z6			
14															Z4					
15																				
16																				
17						Z7														
18																				
19		Z8																		
20																				

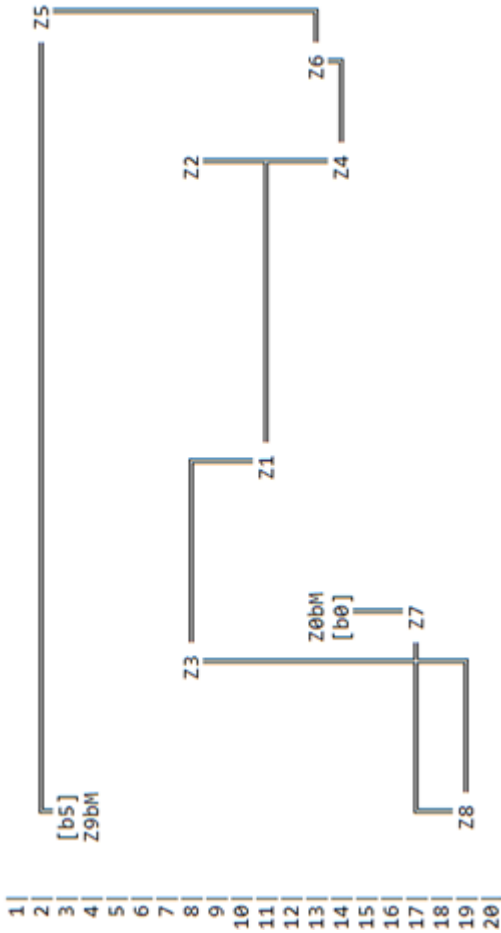
**Annexe (B) :** Carte géographique après le placement des bus et rues, et l'enlèvement de l'affichage : les lignes de tirets et les barres verticales entre les cases.

bM : bus multiple (plusieurs bus qui stationnent dans cet endroit).



Les annexes de (C) à (H) représentent le traçage de la carte géographique à double affichage dans des minutes différentes : affichage normal de circulation de bus (à gauche) et affichage de circulation de bus et du nombre de personnes par zone et bus (à droite).

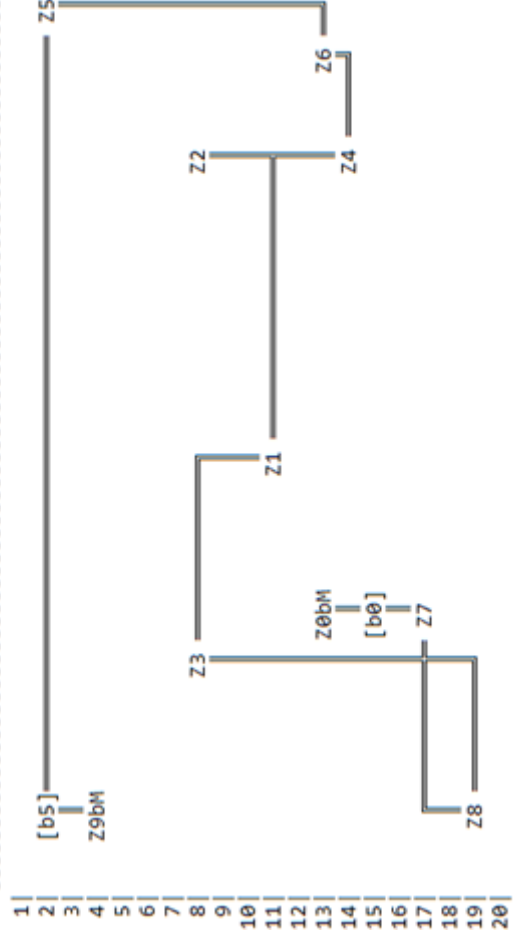
now (J1) 6H:1m  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20



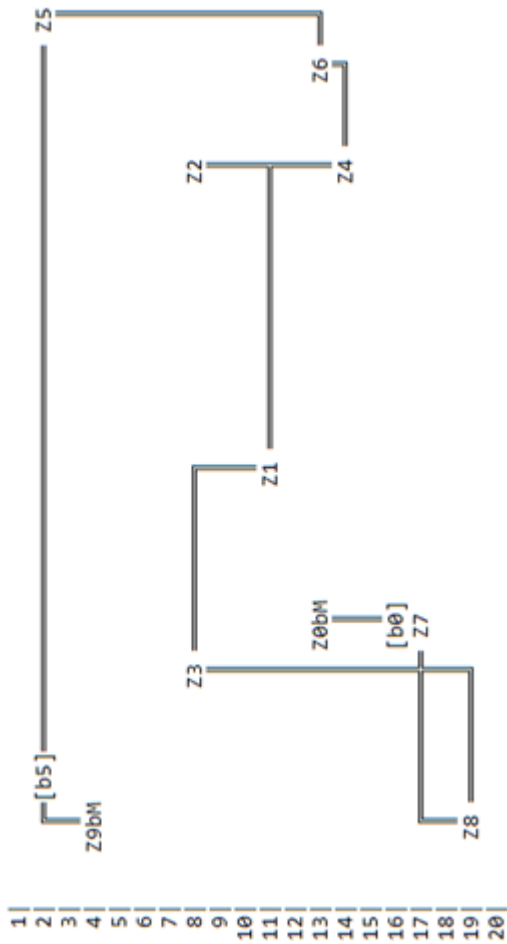
Annexe (C)

Annexe (D)

now (J1) 6H:2m  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20



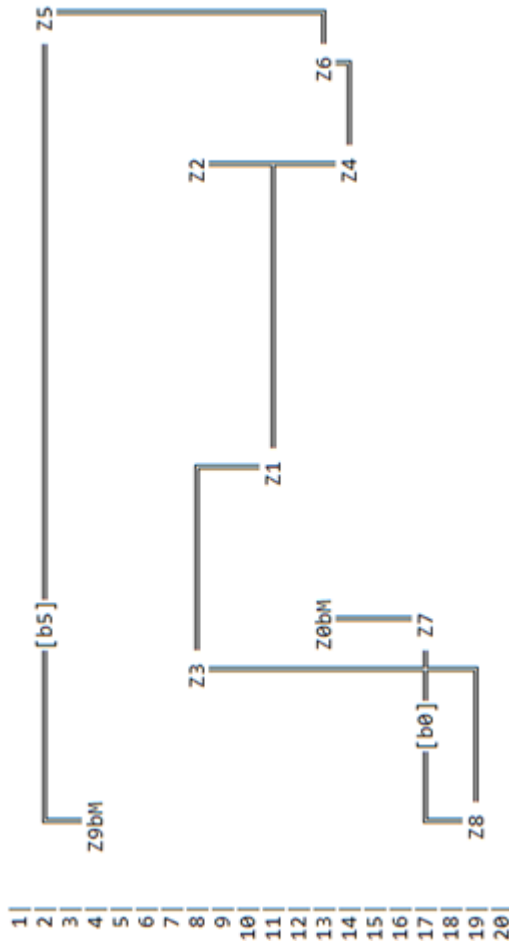
now (J1) 6H: 3m  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

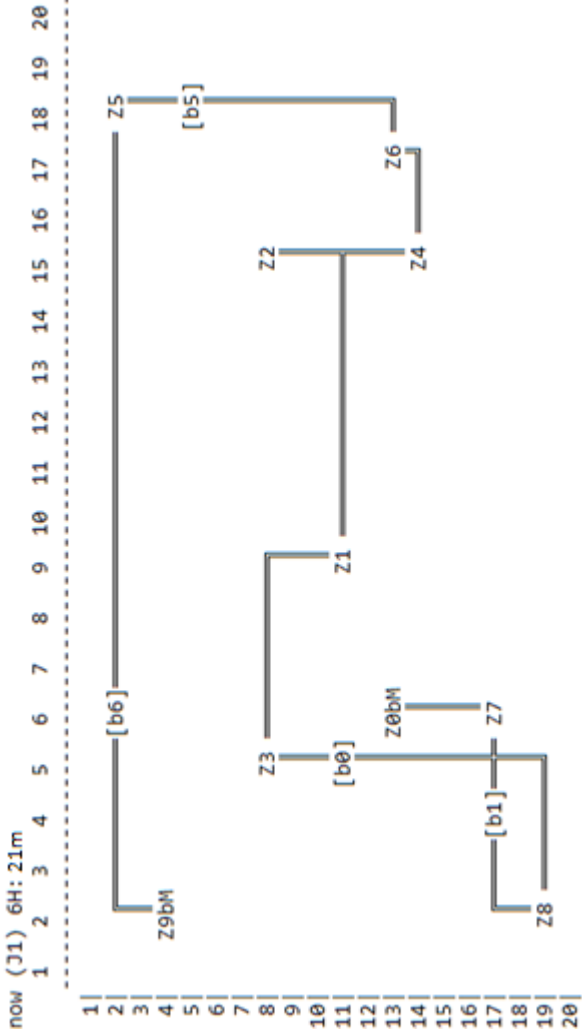


Annexe (E)

Annexe (F)

now (J1) 6H: 6m  
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20





Annexe (G)

Annexe (H)

