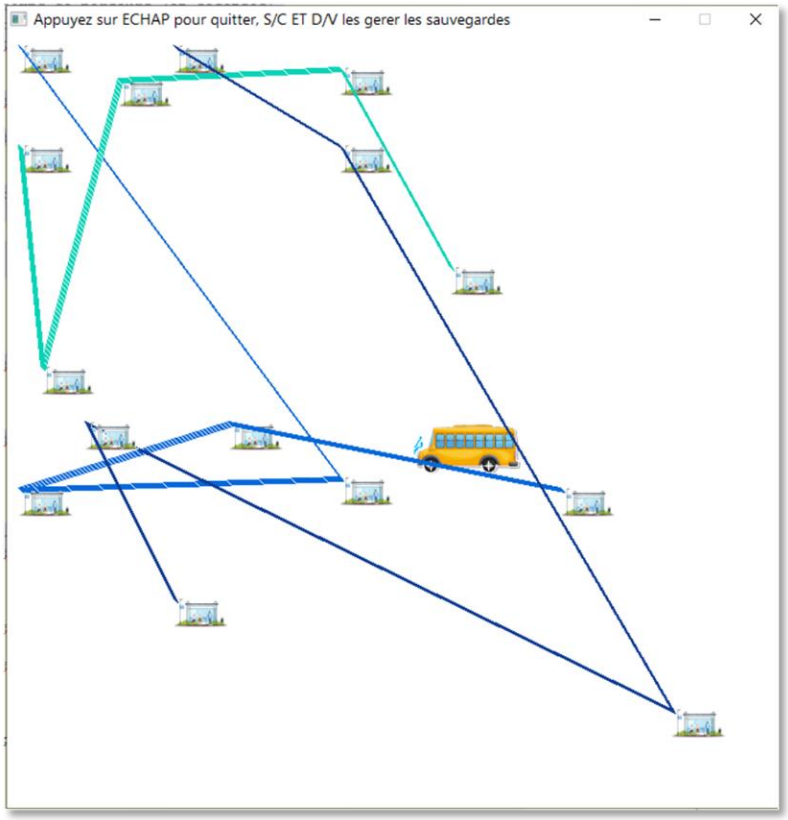
Projet bus

L'idée est de créer un programme qui simule un ou plusieurs bus se déplaçant sur une ou plusieurs lignes de bus.



Les types utilisés

Tous les types liés à la simulation (hors interface graphique) sont définis dans types.h.

Une ligne de bus

Une ligne de bus est composée de stations et de tronçons entre deux stations.

Une station et un tronçon seront vus comme un type Tstation, où certains champs seront utiles soit à l'un, soit à l'autre.

Voici sa définition dans types.h :

```
typedef struct T_station { //représente UN TRONCON (entre deux arrêts) ou UN ARRET (cad une station de la ligne de bus)

  TypeNoeud arret_ou_troncon;

  //données utiles si le Noeud est un TRONCON
  int idLigneBus;              //numéro de la ligne de bus par laquelle on arrive sur la station
  struct T_station* depart;       // pointeurs sur les stations de départ et d'arrivée du tronçon
```

```
   struct T_station* arrivee;
   int coutTemps;                // coût en temps de parcours (en secondes)
   int coutDistance;             // coût en distance (en mètres)
   int tempsCumule, distanceCumule;   // champs utiles pour déterminer le meilleur itinéraire

   //données utiles si le Noeud est un ARRET
   int posX, posY;               //coordonnées sur le plan de la ville
   char nomStation[30];
   int idStation;                //numéro unique et non lié à une ligne, une ligne de bus étant constituée d'une suite d'id stations: exemple : 3  - 7 - 1 - 5

   //champs utiles pour les tris
   //int coutMaintenance;            //en Kilo euros, une valeur entre 10 et 100  (A VOUS DE GERER)
   //t_date dateDerniereMaintenance       //t_date est à définir, doit permettre de stocker une date jour/mois/année   (A VOUS DE GERER)

   //struct T_station* correspondances; //égale à NULL si pas de correspondance, sinon égale à une liste de TRONCONS
   //int coutCumule;               //pour le calcul du plus court chemin, algo de Djikstra

} Tstation;
```

Les stations seront stockées dans des listes doublement chainées**, <u>via leur adresse</u>**, dans le champ pdata :

```
//type pour la liste doublement chainée, utilisé dans listeDouble.h/.c
typedef struct T_cell{
   struct T_cell *suiv;
   struct T_cell *prec;
   Tstation *pdata;
} T_cellule;
typedef T_cellule* T_liste;
```

Pour masquer cette liste de base et pour coller à ce qu'elle représente dans la librairie qui va gérer les lignes de bus, nous encapsulons T_liste (listeDouble.h/.c) dans le type TlisteStation (ligneBus.h/c).

```
//Une liste de stations  (basée sur listeDouble.h/.c)
typedef T_liste TlisteStation;  //T_liste et TlisteStation sont équivalents, cad pointeur sur Tcellule
```

Aussi, il faut bien avoir en tête qu'un TlisteStation est une liste de Tcellule avec comme donnée une adresse sur un Tstation, <u>commençant et se terminant par une STATION</u>. Entre le départ et le terminus, nous trouverons <u>une alternance de STATION/TRONCON</u> !

La fonction :

```
TypeNoeud getTypeNoeud(Tstation *myStation);
```

présente dans types.h/.c nous permettra de savoir à quoi nous avons affaire à chaque Tstation.

Un bus

Un type Tbus, pointeur sur une structure, nous permettra de gérer un bus. Les getteurs et setteurs associés sont fournis dans types.h/.c.

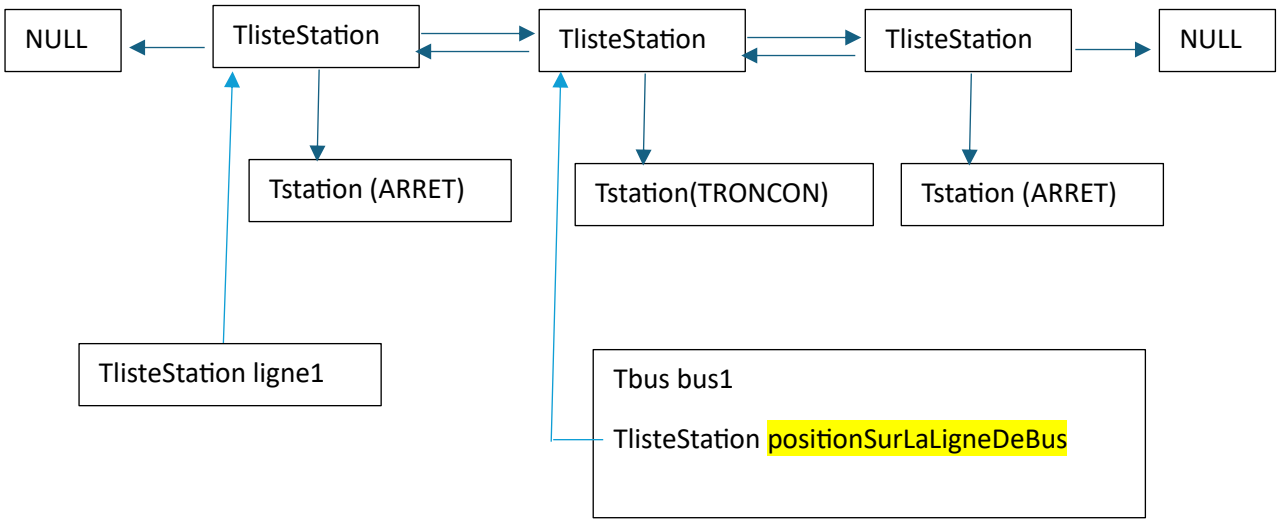
On remarquera le champ positionSurLaLigneDeBus, qui est un pointeur sur une cellule de la liste doublement chainée de type TlisteStation, pour indiquer où est le bus sur la ligne.

```
typedef struct{
   int idBus;
   TlisteStation positionSurLaLigneDeBus;  //la derniere station atteinte (dans la liste) du bus est dans le champ pdata, stocké dans un Tcellule
   int posXBus, posYBus;              //coord du bus pixel par pixel pour les trajets
   int idLigneBusActuelle;            //un bus suit une ligne de bus à un instant t et peut en changer
   TsensParcours sensParcours;
} Typebus;

typedef Typebus *Tbus;
```

Schéma en mémoire

Voici en image une ligne de bus composée de deux stations et d'un bus la parcourant :



Point de départ :

Une archive « projetBus.zip » vous est fournie sur updago. Commencer par regarder types.h/.c, le main… La partie interface graphique est « transparente » pour vous (maSDL.h/.c et SDL_VSYNC.h/.c). Elle utilise la SDL version 2 et les options de compilation sont présentes dans les options du projet codeblock (clic droit sur « projetBus » dans le manager de projet, item « build options », onglet « linker setting ») si vous êtes curieux.

Critères de notation :

- La complexité en temps et en espace de vos fonctions
- L'utilisation de l'encapsulation
- Vos arguments pour expliquer vos choix
- La lisibilité de votre code

Votre travail :

Est à faire en binôme exclusivement. La date de dépôt sera fixée ultérieurement.

1. Getteurs et Setteurs

- ajouter les champs coutMaintenance et dateDerniereMaintenance dans le type Tstation, dans types.h. Ces deux champs devront être initialisés avec des valeurs aléatoires.

```
  //champs utiles pour les tris
  //int coutMaintenance;            //en Kilo euro, une valeur entre 10 et 100  (A VOUS DE GERER)
  //t_date dateDerniereMaintenance      //t_date est à définir, doit permettre de stocker une date jour/mois/année   (A VOUS DE GERER)
```

- créer les getteurs/setteurs associés

2. Tris

On souhaite trier les stations et tronçons d'une même ligne de bus sur leur coût de maintenance, de façon **décroissante**.

On souhaite également pouvoir trier les stations par date de maintenance, de façon **croissante**.

Implanter **une (seule) fonction de tri** qui servira pour les deux tris précédents et **les fonctions de comparaisons** utiles.

On souhaite maintenant ne prendre en compte que les tronçons ou que les stations dans le résultat des tris. Proposer une solution (que vous mettrez en application dans le main via une touche lue dans la boucle principale du main qui déclenchera son appel).

Créer une librairie pour votre (unique) fonction de tri. À vous de gérer les « include » nécessaires.

3. Améliorer TlisteStation creeLigneDeBus1()

Dans les fonctions creeLigneDeBus1/2/3 (ligneBus.c), une chose ne va pas : les valeurs fournies aux appels de « creeTroncon », ici 35 et 280 dans l'exemple qui suit.

troncon = creeTroncon(1,dep,arr,**35**,**280**);

On souhaite que ces informations soient calculées par rapport aux coordonnées en X et Y des stations dep et arr : les coûts en temps et en distance seront les mêmes (pour l'instant) et égaux à la distance entre les stations.

Proposez et appliquez une solution.

4. Gestion de fichiers (version1)

À partir des informations présentes dans les fonctions creeLigneDeBus1/2/3 (ligneBus.c) et en tenant compte de votre amélioration liée à la question précédente, créer un fichier séquentiel "**Stations et lignesDeBus.data**" qui sera lu au démarrage du programme et permettra la création des lignes de bus.

Les fonctions **creeLigneDeBus1/2/3** ne sont là que pour la démonstration initiale du projet et doivent être remplacées par une solution qui charge « **Stations et lignesDeBus.data »**

Vous devez également imaginer une sauvegarde de la position du Bus dans ce fichier.

5. Créer les fonctionnalités suivantes

Pour chacune des fonctions suivantes, vous devez les appeler dans votre « main.c » d'une façon ou d'une autre pour la démonstration de votre code le jour de la soutenance.

1. Une fonction permettant de construire une ligne de bus en reprenant les stations de deux autres. Le départ sera celui de la première et le terminus sera celui de la seconde.
2. Une fonction permettant la suppression d'une station d'une ligne de bus (démonstration à chaud svp).
3. Une fonction qui rend circulaire une ligne de bus : la station suivant le terminus est la station de « départ ». Trouvez un moyen de créer un compteur de tour et d'afficher cette information dans la console.

6. Plusieurs bus

Proposer une solution pour que plusieurs bus circulent en même temps. Commencer par regarder comment fonctionne bus1, dans le « main ».

7. Une même station sur plusieurs lignes

Proposer une solution pour qu'une même station appartienne potentiellement à plusieurs lignes de bus. Si besoin, proposer une version 2 des fichiers de sauvegardes.