

**Gestion d’un réseau de tramways**

HENZELIN Camille

HUG Laurine

SCHERRER Xavier

WEISSENBERGER Julien

FST 2017/2018

Février/Mars 2018

Sommaire

[**I-** **Introduction** 3](#_Toc509782839)

[**II-** **Fonctionnement du programme** 4](#_Toc509782840)

[**III-** **Organigramme** 8](#_Toc509782841)

[**IV-** **Choix d’organisation** 8](#_Toc509782842)

[**V-** **Contribution** 9](#_Toc509782843)

[**VI-** **Conclusion** 10](#_Toc509782844)

1. **Introduction**

Le projet qui nous a été donné consistait à développer un programme qui permet de simuler des lignes de tramways grâce à des informations donné au moyen de fichiers externes pouvant être modifiés directement à la main.

Pour développer notre programme nous avons utilisé le logiciel CodeBlocks. Nous nous sommes également servis de winBGI afin de coder notre fenêtre graphique pour représenter les tramways sur nos différentes lignes et différents arrêt.

1. **Fonctionnement du programme**

Notre fonction main appelle la fonction chargerLignes. Cette fonction va récupérer les données du fichier externe afin de créer les lignes.

Le fichier externe contient toutes les informations des lignes :

- Le nombre d’arrêt qu’elles contiennent

- Le nom, la durée de l’arrêt, la position en x et la position en y des arrêts

- Le nombre de tramways

- Le numéro, la vitesse maximal, la vitesse, la distance minimale, le temps restant à l’arrêt, la distance jusqu’au prochain arrêt, le sens de déplacement et le nom du prochain arrêt pour chaque tramways

ListeLigne :

La classe listeligne contient un constructeur par défaut qui l’initialise à « nullptr », ainsi qu’un destructeur qui va supprimer tous les chaînons présent dans la liste.

Son constructeur par recopie va permettre de créer une nouvelle liste de ligne à partir d’une liste déjà existante.

La fonction « ajouter » permet de d’ajouter un nouveau chaînon dans la liste chaînée.

La fonction « affiche » permet d’afficher les arrêts et les tramways dans la fenêtre graphique

La fonction « rafraichir » permet de gérer le déplacement des tramways

Les chaînons de cette liste sont composés de différentes lignes.

Ligne :

Les lignes sont des classes composées de listeArret et listeTramways contenant respectivement les différents arrêts et tramways dans une liste chaînée.

Elle contient un constructeur par défaut qui initialise la liste avec « nullptr ».

Cette classe possède également un constructeur par recopie afin d’initialiser les données de la nouvelle avec celles d’une liste déjà existante.

« ajouterArret » et « ajouterTram » permettent d’ajouter respectivement un nouvel arret et un nouveau tramway à la ligne.

« chercherArret » et « chercherTramway » permettent de rechercher dans la ligne un arrêt et un tram en donnant respectivement le nom de l’arrêt et le numéro du tram.

La fonction « afficheArret » fait appel à « affiche » de la classe arret.

La fonction « GetPosArret » permet l’affichage des lignes qui lient les arrêts.

La fonction « afficheTramway » permet d’afficher les tramways.

La fonction « affiche » gère l’affichage total de la ligne.

ListeArret :

La classe listeArret possède un constructeur par défaut qui l’initialise à « nullptr ».

Son constructeur par recopie permet de créer une nouvelle liste en réutilisant les données d’une liste passée en paramètre.

Son destructeur va supprimer chaque chaînon de la liste jusqu’à ce qu’elle soit vide.

Cette classe est composée de plusieurs classe arret sous forme de liste chaînée. Elle contient plusieurs fonctions, tel que « ajouter » qui permet d’ajouter un nouveau chaînon arret dans la liste en le plaçant en fonction de l’ordre sur la ligne.

La fonction « supprimer » permet d’enlever un chaînon de la chaîne. Le chaînon est trouvé grâce au nom de l’arrêt passé en paramètre.

La fonction « chercher » va rechercher un arrêt dans la liste chaînée en comparant les noms des arrêts avec la chaîne de caractère passé en paramètre puis la renvoyé si il se trouve dans la liste.

Arret :

La classe arret contient les données d’un arrêt :

-le nom

-la durée d’arrêt

-la position

-un pointeur sur le suivant

-un pointeur sur le précédent

Son constructeur permet d’initialiser ces données grâce aux différentes variables passées en paramètre.

Elle est utilisée comme chaînon par la classe listeArret.

En plus de son constructeur, elle contient la fonction affiche qui permet d’afficher la position de l’arrêt à l’écran.

ListeTramways :

La fonction ListeTramways possède un constructeur par défaut qui initialise la tête de liste à « nullptr ».

Elle possède aussi un constructeur par recopie qui prend les données d’une autre liste de tramways.

Son destructeur supprime tous les tramways de la liste en partant de la tête.

ListeTramways ajoute des tramways à sa liste grâce à la fonction « ajouter » qui crée un tramway avec les données requises (numéro du tram, ligne, vitesse maximale, vitesse, distance minimum entre tram, temps d’arrêts, distance prochain arrêt, sens de déplacement, arrêts suivant, arrêts précédent)

ListeTramways peut aussi « supprimer » un tramway à l’aide de son numéro de tram.

On peut obtenir les informations d’un tramway en particulier en le cherchant avec son numéro de tram avec la méthode « chercher »

La fonction « affiche » permet l’affichage de la liste de tramways

La fonction « rafraichir » gère le déplacement des tramways

Tramway :

La classe Tramway est la classe utilisé par ListeTramway pour lister les véhicules Ferroviaires.

Son Constructeur par défaut construit un tram avec les données décrites précédemment.

On peut calculer à l’aide de la classe Tramway et sa fonction « distance »:

-la distance entre deux tramways

-la distance entre deux arrêts

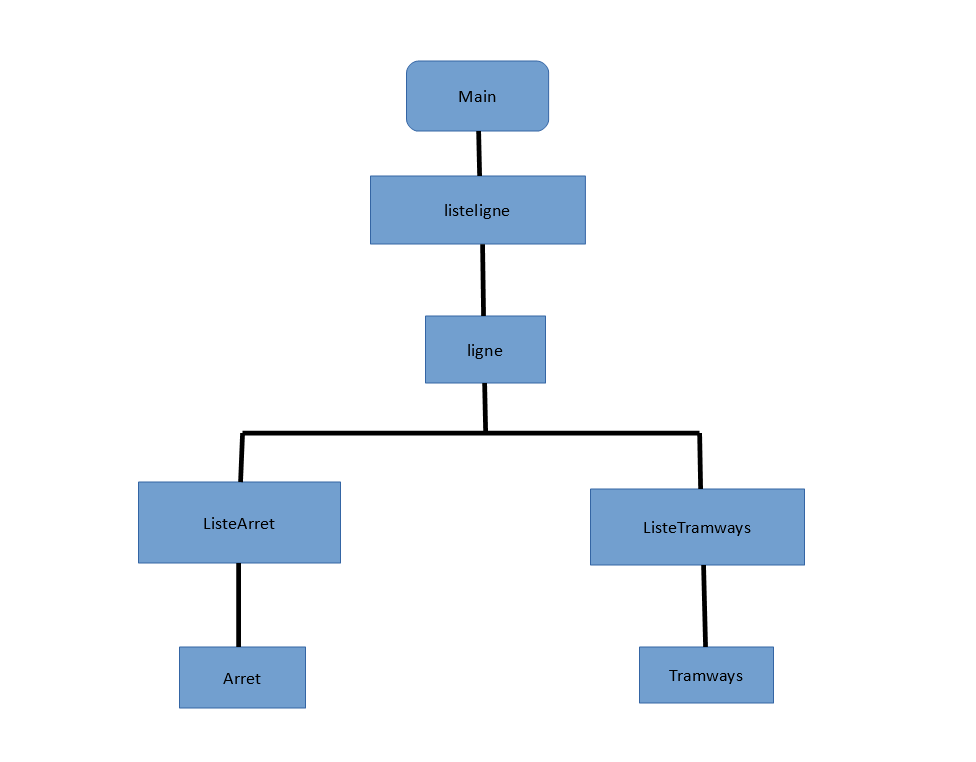
-la distance entre un tramway et un arrêt

« GetPosition » donne la position d’un tram.

« affiche » permet l’affichage d’un tramway

« avancer » permet de faire avancer le tramway

1. **Organigramme**



1. **Choix d’organisation**

Pour la lecture des fichiers nous avons décidé de mettre nos données dans un seul fichier plutôt que de les séparer. Nous avons fait cela afin de facilité la lecture sans devoir chercher les différents fichiers correspondant à celui que l’on recherche.

L’utilisation des listes doublement chaînées a permis une facilité d’utilisation pour l’organisation des listes, notamment pour la liste des trams afin de retrouver le tramway devant et derrière lui.

Nous avons décidé d’utiliser WinBGI pour notre fenêtre graphique car nous avions déjà dû l’utiliser lors de TP. Nous savions donc déjà nous en servir et ce programme est facile d’utilisation.

1. **Contribution**

Camille Henzelin :

Le constructeur et « affiche » de la classe « arret » ont été écrits par Camille.

Elle a aussi aussi écrit les fonctions : « chercherArret », « chercherTramway », « afficheArret », « afficheTramways » et « affiche » de la classe « ligne ».

Les fonctions « chercher », « CoordsArrets » et « afficher » ont été implémentées par Camille.

Camille a écrit les fonctions « chercher », « afficheArrets » et « affiche » de la classe listeTramways.

Le constructeur et les fonctions « affiche » et « GetPosition » de la classe « tramway » ont été faits par Camille.

Laurine Hug:

Laurine s’est occupée en partie de la classe listeArret. Elle a programmé le destructeur de la classe ainsi que les fonctions « ajouter » et « supprimer ».

Elle a également écrit le constructeur par défaut, par recopie et le destructeur de la classe listeligne.

Julien Weissenberger :

Julien a codé des fonctions de la classe listeTramways. Il a codé le constructeur par défaut, le constructeur par recopie ainsi que le destructeur. Les fonctions « ajouter » et « supprimer » ont également été faites par Julien.

Il a fait les constructeurs par recopie et par défaut de la classe « ligne ».

Julien a écrit les fonctions « rafraichir » de la classe « listeligne » et « listeTramways ».

Les trois fonctions « distance » et la fonction « avancer » de la classe « tramway » ont été implémentées par Julien.

Xavier Scherrer :

Xavier s’est occupé de la fonction « ajouter » de la classe listeligne.

« GetPosArret », « ajouterArret » et « ajouterTram » de la classe ligne on aussi été faites par Xavier.

La fonction « affiche » de la classe « listeligne » a été écrite par Xavier.

Le constructeur par défaut et par recopie de la classe « listeArrets » ont été implémentés par Xavier.

1. **Conclusion**

Nous avons trouvé ce projet très intéressant, il nous a permis de mettre en pratique la programmation des listes chaînées. Grâce à cette mise en pratique nous avons eu l’occasion de mieux comprendre leur fonctionnement.

Durant le projet nous avons rencontré de nouvelles erreurs. Devoir les corriger nous a permis d’améliorer nos connaissances en programmation.

Nous avons aussi appris à organiser notre programmation, répartir le travail entre les différents membres du groupe et prévoir des réunions pour mettre en commun ce que nous avons fait.

Pour conclure ce projet a été très bénéfique pour nous.