Document accompagnateur

Description Explicative du projet «Voisin»

Maxime Laurin

Département d’informatique

Cégep de Victoriaville

Janvier 2024

*Première demande du professeur :*

* Un document accompagnateur décrit et justifie les choix effectués lors de votre développement de la classe **ResourceLoader**et de votre implémentation du chaînage

Le développement de ResourceLoader c’est fait avec beaucoup à tâtons et surtout d’apprentissage. Ma première version, je séparais toutes les valeurs pour les mettre dans un vector. Une classe qui contient 12 vector pour chaqu’une des catégories.

|  |
| --- |
| class ResourceLoader  {  Vector<double> acide\_fixe  Vector<double> acide\_volatile  [ … ]  }; |

Mais ça me demandait d’utiliser de la séparation un peu plus boiteuse que j’ai n’ai pas entièrement compris. Pour améliorer mon code, j’ai cherché comment le replacé, j’ai donc tenté un moyen pour créer un vector pour chaque ligne mais le problème revenait un peu au même et qu’il sera complexe de tout mettre ensemble. Après explication de M.Frédérik Taleb sur le placement chainer, j’ai décider d’adapter ma classe malgré que cela impliquais de modifier presque toute les méthodes de la classe. J’ai donc modifié mon code pour agir selon le principe des Nœuds et du chainage.

ResourceLoader : Nœud:

|  |  |
| --- | --- |
| class ResourceLoader  {  private:  Noeud\* m\_premier;  public:  [Méthodes]  }; | struct Noeud  {  vector<double> valeurs;  Noeud\* suivant;  }; |

J’ai mis en place les valeurs dans un seul vector afin de faciliter l’utilisation de la méthode « CalculerEuclidienne » qui prend deux vector de taille n et qui calcule la distance Euclidienne entre les points des vectors.

*Seconde demande du professeur :*

* À vous de déterminer et justifier la façon d'accéder aux données (enregistrement et attributs d'un enregistrement) dans un **document accompagnateur**.

À l’initiation de la classe, le nœud\* m\_premier sera mis à null, indiquant l’absence des valeurs. Seulement quand la méthode « TraiterFichierCSV » est appeler avec un fichier correspondant les valeurs seront ajouter directement dans le prochain nœud libre de la chaine. Pour chaque données, une vérification du prochain nœud est faite pour savoir quand s’arrêter pour ajouter le nœud et ses données.

|  |
| --- |
| Noeud\* nouvelleValeur = new Noeud{  donnees,  nullptr  };  if (m\_premier == nullptr) {  m\_premier = nouvelleValeur;  }  else {  Noeud\* iterateur = m\_premier;  while (iterateur->suivant != nullptr) {  iterateur = iterateur->suivant;  }  iterateur->suivant = nouvelleValeur;  } |

Si la classe est supprimer, un itérateur va être créer pour commencer au début de la chaine. S’il y a un prochain, il va copier l’adresse du nœud qui commence la chaîne, se déplacer au 2e et un itérateur supplémentaire va rester avec le premier pour que les deux se supprime ensemble.(Voir Schéma en haut de la prochaine page)

|  |
| --- |
| ResourceLoader::~ResourceLoader()  {  if (m\_premier == nullptr)  {  delete m\_premier;  }  else  {  Noeud\* iterateur = m\_premier;  while (iterateur != nullptr)  {  Noeud\* aSupprimer = iterateur;  iterateur = iterateur->suivant;  delete aSupprimer;  }  }  } |