DUT Informatique Graphique
Le Puy en Velay

Graphe
TP 1

Imagerie 1 — Pierre-Yves Bischoff — 2021

Programmation par tests, création d'une application de gestion de graphe

Nous allons utiliser les 2h de TD/TP plus la première séance de TP pour faire ce TP.

Exercice 1 : Rappel

Un graphe est composé de deux éléments centraux :

1. Une liste ou un set de nœuds, pouvant porter une information (un nom, un identifiant, etc.),
2. Une liste ou un set d'arêtes, pouvant être orientées et pondérées.

Pour modéliser un graphe, nous allons concevoir à travers une série de TP plusieurs classes permettant de formaliser et exploiter ces concepts.

Exercice 2 : Création du modèle : Application C + + console

Un modèle de graphe simple est défini à l'aide de trois classes : `CGraph`, `CNode`, `CEdge`.

- Pour l'instant, un `CNode` ne contiendra qu'une information de type `std::string` pour son nom.
- Un `CEdge` relie deux `CNodes`, et peut être pondéré.
- Un graphe doit permettre :
  1. d'obtenir le nombre de sommets et le nombre d'arêtes présents dans le graphe,
  2. d'ajouter sommets et arêtes,
  3. de supprimer un sommet ou une arête,
  4. de trouver un sommet par valeur (nom) ou par référence,
  5. de parcourir tous les sommets, toutes les arêtes,
  6. d'obtenir le degré d'un sommet,
  7. d'afficher à l'aide de l'opérateur chevron (<<) les informations sur les nœuds et arêtes le composant.
- Il doit également s'assurer de son intégrité :
  1. Lors d'un ajout d'arête, vérifier que les nœuds existent,
  2. Lors de la suppression d'un nœud, supprimer les arêtes y faisant référence.

Exercice 3 : Le travail à faire

Je vous conseille pour ne pas vous perdre dans votre programmation de tout programmer dans les fichiers header de chaque classe. Je vous joins un squelette qui compile.

1. Programmez la classe CNode.
2. Ensuite vous ferez la classe CEdge.
3. Enfin vous finirez par la classe CGraph.

Exercice 4 : Vérification du modèle du graphe

Pour valider votre programmation du modèle du graphe, vous devrez enlevez les commentaires qui libèrent certains tests au fur et à mesure.

1. Créer un modèle vide, tester si les fonctions donnant le nombre de sommets et d'arêtes renvoient bien la valeur zéro.
2. Ajouter deux sommets, vérifier les mêmes fonctions que précédemment avec les nouvelles valeurs attendues.
3. Ajouter une arête, vérifier encore le nombre d'arêtes disponibles.
4. Ajouter un nouveau sommet et deux arêtes pour obtenir un graphe complet d'ordre 3.
5. Vérifier le résultat de la fonction permettant d'obtenir le degré d'un nœud.
6. Supprimer le dernier sommet ajouté. Vérifier que les compteurs de sommets et d'arêtes sont bien à jour.
7. Vérifier également que la fonction permettant de trouver un sommet par son nom fonctionne avec l'un des sommets initiaux, et renvoie un pointeur vide pour un sommet n'existant pas dans le graphe.
8. Et pour terminer, vérifier le résultat de l'opérateur <<.

Exercice 5 : Optionnel!!! Création d'un arbre

Rappel : un arbre est un graphe particulier qui possède un élément principal : la racine.

1. Rajoutez la classe $CTree$ qui dérive de $CGraph$.
2. Ajouter une fonction $enraciner$ qui enracine un arbre au sommet s.
3. Ajouter une fonction $hauteur$ qui renvoie la hauteur de l'arbre.
4. Créer l'opérateur $<<$ qui affiche l'arbre par hauteur croissante.