

Construction d'un ensemble de routines pour la gestion d'un graphe. – Programmation en C++ -

Introduction

On se propose de programmer un ensemble de routines de base (structures, types, procédures et fonctions) pour la gestion d'un graphe (orienté et non orienté). Le langage de programmation utilisé est le C++.

Afin de simplifier le travail, nous ne considérons que des graphes composés de sommets dont les noms sont composés des lettres de l'alphabet en majuscule ou minuscules : 'A', 'B', 'C', ... 'Z', 'a', 'b', 'c', ... 'z'. Avec cette hypothèse, vous pouvez travailler sur des graphes dont le nombre maximum de sommets est de 52 ($2*26$)

I – Définition des types de base pour la construction d'un graphe

Pour construire un graphe, on définit en C++ les éléments suivants :

```
// Définition du type correspondant à la matrice d'adjacence du graphe
typedef int mat[52][52];
```

```
// Définition d'un arc ou d'une arête
struct arc_arete
{
    char initial; // sommet initial
    char final; // sommet final
    float poids ; // cout de l'arc ou arête
}
```

```
// Définition d'un graphe
struct graphe
{
    int nature_du_graphe; // valeur 0 si graphe orienté et valeur 1 si graphe non orienté
    int nb_sommets; // nombre de sommets du graphe
    int nbarcs; // nombre d'arcs du graphe
    char liste_sommets[52] // liste des sommets du graphe
    arc_arete arcs[100]; // liste des arcs ou arêtes du graphe (100 au maximum)
    mat matriceadjacence; // matrice d'adjacence du graphe
}
```

Écrire et tester les procédures suivantes :

1 - Initialisation de la matrice d'adjacence d'un graphe (tous les éléments sont mis à 0)

void init_graphe(graphe &g)

2 - Construction d'un graphe à partir d'éléments saisis au clavier

void saisie_graphe (graphe &g)

3 - Affichage de la matrice d'adjacence d'un graphe

void afficher_matrice_graphe (graphe g)

4 - Affichage des sommets d'un graphe

void afficher_sommets_graphe (graphe g)

5 - Affichage des arcs ou arêtes d'un graphe

void afficher_arcs_ou_arettes(graphe g)

6 - Afficher les sommets adjacents d'un sommet. Dans le cas d'un graphe orienté préciser s'il s'agit de successeurs ou prédécesseurs

void sommets_adjacents (graphe g, char sommet)

7 - Calcul du carré de la matrice d'adjacence d'un graphe

void carre_matadj_graphe (mat m, mat &carre_de_m);

II – Chargement d'un graphe à partir d'un fichier texte

La saisie des informations d'un graphe à partir du clavier peut être une opération très fastidieuse (imaginez un graphe avec un nombre de sommets et d'arcs élevé). Pour remédier à cela, il vous est demandé de réaliser la saisie et la construction d'un graphe à partir de données stockées dans un fichier texte appelé : ***graphe.dat***.

Afin de manipuler les données stockées dans le fichier *graphe.dat*, utilisez le flot d'entrée sortie du langage C++, définie par :

. Importer la bibliothèque *fstream.h* (nécessaire pour la gestion de fichier)

#include <fstream>

. Ouverture du fichier *fichier* en lecture et assignation au fichier physique « *graphe.dat* »

ifstream fichier (« graphe.dat »);

// lecture d'une donnée courante à partir du fichier *fichier* et stockage dans la variable *x*

fichier >> x;

. fermeture du fichier à la fin des traitements

fichier.close g();

9 - Réécrire la procédure de saisie du graphe à partir de données stockées dans le fichier texte

void saisie_graphe_bis (graphe &g);