TP Graphes 4

Licence Informatique - 2nde année

Année 2023-2024

1 Compilation séparée

Dans ce premier exercice, vous allez être amenés à expérimenter la compilation séparée, à partir de la correction qui vous est fournie du TP3. Le fichier `tp3_corrige.cpp` qui se trouve dans le sous-dossier `TP4/Exercice1`, contient en effet toutes les fonctions liées à la fois aux matrices et au parcours en largeur du graphe. L'objectif sera alors de séparer les sources en quatre fichiers distincts, contenant respectivement le programme principal (fichier `tp4.cpp`), les fonctions liées aux matrices (fichier `matrices.cpp`), les fonctions de gestion d'une file de type FIFO (fichier `fifo.cpp`) et enfin celles liées au parcours (fichier `parcours.cpp`).

Préliminaire

Après avoir recopié (ou renommé) la correction fournie dans un fichier nommé `tp4.cpp` (qui devra se trouver dans le sous-dossier `TP4/Exercice1`), réalisez chacune des étapes qui suivent.

Module `fifo.cpp`

Créez le fichier `fifo.cpp`, qui contiendra toutes les fonctions liées à la gestion d'une file de type FIFO, qui se trouvent actuellement dans `tp4.cpp`. A l'issue de cette étape, la définition de ces fonctions ne devra plus être présente dans ce fichier.

Remarque : il sera nécessaire d'inclure le fichier `types.hpp` dans `fifo.cpp` (directive `#include`), de manière à ce que le compilateur connaisse les types à utiliser lors de la compilation de ce module.

Fichier `fifo.hpp`

En vous appuyant sur ce qui a été vu en cours, créez le fichier `fifo.hpp`, qui contiendra la définition des fonctions exportées par le module `fifo.cpp`. Mettez à jour en conséquence le fichier `tp4.cpp`, en y supprimant la prédéfinition des fonctions liées aux FIFO et en y incluant le fichier `fifo.hpp`.

Fichier `Makefile`

En vous appuyant sur ce qui a été vu en cours, créez le fichier `Makefile` de votre application. Utilisez ce fichier pour compiler cette application et vérifier son bon fonctionnement.

Module `matrices.cpp`

Créez le fichier `matrices.cpp`, qui contiendra toutes les fonctions liées aux matrices qui se trouvent dans `tp4.cpp`. A l'issue de cette étape, la définition de ces fonctions ne devra plus être présente dans ce fichier.

Remarque : comme précédemment, il sera nécessaire d'inclure le fichier `types.hpp`, de manière à ce que le compilateur connaisse les types à utiliser lors de la compilation de ce module. Il sera également nécessaire d'inclure les bibliothèques `iostream` et `fstream`, dont certaines fonctionnalités sont utulisées pour l'affichage et l'accès aux fichiers.

Fichier `matrices.hpp`

Créez à présent le fichier `matrices.hpp`, qui contiendra la définition des fonctions exportées par le module `matrices.cpp`. Mettez à jour en conséquence le fichier `tp4.cpp` et le fichier `Makefile` puis recompilez et testez l'application.

Fichiers `parcours.cpp` et `parcours.hpp`

De même que précédemment, créez les deux fichiers liés aux fonctions de parcours d'un graphe (module et fichier d'export), mettez à jour `tp4.cpp` et le fichier `Makefile` de l'application, compilez et testez celle-ci.

Modifications

— Ajoutez une ligne de code au début de la fonction `main`, qui affiche à l'écran vos nom et prénom. Recompilez l'application et vérifiez que seul le fichier `tp4.cpp` est recompilé, en plus de l'étape d'édition de liens.
— Ajoutez une ligne de code dans la fonction `charger` du module `matrices.cpp` qui affiche le nom de la matrice en cours de chargement. Quel(s) fichier(s) doi(ven)t être recompilé(s) ? Vérifiez le en lançant la commande de compilation.

2 Parcours en profondeur

Préliminaire

Recopiez dans le sous-dossier `TP4/Exercice2` le contenu du sous-dossier `TP4/Exercice1`. Modifiez ensuite votre `main` de manière à ce que :
— il n'attende qu'un seul paramètre qui sera le nom du fichier contenant la matrice d'adjacence ;
— il ne contienne que le chargement et l'affichage de la matrice.

Version récursive

En vous appuyant sur ce qui a été vu en cours sur le parcours en profondeur, implantez les fonctions qui permettent de l'implanter dans le fichier `parcours.cpp`. Vous modifierez ensuite la fonction `main` pour y tester uniquement ce parcours. Vous trouverez en annexe les résultats à obtenir pour les différentes matrices fournies dans le dossier `Data`.

Version non récursive

La version vue en cours utilise la récursivité pour le parcours en profondeur. On souhaite remplacer celle-ci par l'utilisation d'une pile, qui contiendra les différents sommets en attente pour la remontée dans le parcours.

Question 1

On propose la structure de pile d'entiers suivante, qui sera représentée par une liste chaînée de `MaillonPile` :

```
struct MaillonPile {
  int valeur; // valeur du maillon
  MaillonPile *dessous; // élément inférieur de la pile
};

struct Pile {
  MaillonPile *sp; // sommet de la pile
};
```

Après avoir ajouté ces structures dans le fichier `types.hpp`, créez un module nommé `pile.cpp` pour y développer les structures de manipulation de pile suivantes :
— `void initialiserPile(Pile &p)` qui initialise la pile passée en paramètre à vide ;
— `bool estVide(Pile p)` qui renvoit `true` si la pile passée en paramètre est vide, `false` sinon ;
— `void empiler(Pile &p, int v)` qui ajoute la valeur `v` en sommet de la pile `p` ;

— `int depiler(Pile &p)` qui enlève le maillon en sommet de la pile `p` et renvoit sa valeur. La pile est supposée non vide ;

Après avoir créé le fichier d'export `pile.hpp` et mis à jour votre fichier `Makefile`, testez ces fonctionnalités au sein de votre `main`.

Question 2

En utilisant cette structure de pile, proposez une seconde version du parcours en profondeur qui n'utilise pas de récursivité et qui sera appelée sous le nom de fonction `parcoursEnProfondeurPile`. On précise les points suivants :

— la pile sera utilisée pour mettre en attente les sommets découverts pendant l'exploration, mais pas entièrement traités (tous les voisins n'ont pas encore été examinés) ;

— pour conserver l'ordre de parcours en profondeur, il ne faut traiter qu'un seul sommet adjacent à la fois pour un sommet qui vient d'être découvert. Associé à la remarque précédente concernant la pile, cela signifie que lorsque vous empilez un sommet, son prédécesseur dans l'ordre de parcours doit être présent dans la pile juste avant lui ...

Il est conseillé de réfléchir au fonctionnement de l'algorithme en le déroulant sur un schéma de graphe simple et en vous appuyant sur les résultats de parcours obtenus par l'algorithme récursif implanté au premier exercice. Une fois réalisée, vous rajouterez cette fonctionnalité dans votre `main` de manière à pouvoir tester les résultats fournis par les deux versions du parcours.

Annexe

Fichier `matrice01.txt`
```
couleurs  : N N N N N
parents   : X 0 1 2 0
0
0 1
0 1 2
0 1 2 3
0 4
```

Fichier `matrice03.txt`
```
couleurs  : N N N N N N
parents   : X 0 1 2 3 4
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
```

Fichier `matrice04.txt`
```
couleurs  : N N N N N N N
parents   : X 0 X 1 3 4 X
0
0 1
2
0 1 3
0 1 3 4
0 1 3 4 5
6
```

Fichier `matrice05.txt`
```
couleurs  : N N N N N N N
parents   : X 0 0 1 3 4 5
0
0 1
0 2
0 1 3
0 1 3 4
0 1 3 4 5
0 1 3 4 5 6
```

Fichier `matrice07.txt`
```
couleurs  : N N N N N
parents   : X 0 1 2 3
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

Fichier `matrice09.txt`
```
couleurs  : N N N N N N N N N N N N N
parents   : X 0 1 1 3 3 3 5 7 8 X 10 11
0
0 1
0 1 2
0 1 3
0 1 3 4
0 1 3 5
0 1 3 6
0 1 3 5 7
0 1 3 5 7 8
0 1 3 5 7 8 9
10
10 11
10 11 12
```