TD de structures de données

Polytech'Lille IMA3

Partie I

Une classe est constituée de NBE élèves. Un élève est représenté par un nom, un prénom, une date de naissance et les NN notes qu'il a obtenues dans les NN matières (1 note par matière).

1. Définir le type Date permettant de représenter une date de naissance
2. Ecrire une fonction qui compare 2 dates d1 et d2 données et retourne :
   0 si d1=d2
   -1 si d1<d2
   1 sinon
3. Définir le type Eleve
4. Ecrire une fonction qui étant donné un élève calcule la moyenne de ses notes
5. Définir le type Classe
6. Ecrire une fonction qui, pour une classe donnée, affiche pour chaque élève : nom, prenom, date de naissance et moyenne
7. Ecrire l'agorithme qui trie les élèves d'une classe par date de naissance croissante, par la méthode du tri bulle.
   Ecrire et utiliser une fonction permuter(eleve1, eleve2) pour réaliser les permutations.

Partie II

1. Définir le type Classe en C
2. Traduire en C la fonction permuter(eleve1, eleve2) et indiquer comment appeler cette fonction dans l'algorithme de tri bulle.

TD de structures de données

Polytech'Lille IMA3

1. Ajouter au type Eleve son numéro NIP unique, sous la forme d'une chaîne de caractères (à 8 chiffres).
2. Montrer que pour optimiser l'accès aux informations d'un étudiant donné par son numéro NIP, il est judicieux que la classe soit ordonnée selon ce critère.
3. On souhaite donner la possibilité d'ajouter/supprimer des élèves dans une classe, dans la limite de MAX_ELEVES. Modifier la structure de données Classe pour offrir cette possibilité.
4. Etant donné une classe ordonnée par numéro NIP et un numéro NIP, écrire la fonction qui renvoie l'indice de rangement de l'étudiant correspondant dans la classe[1].
5. Ecrire l'opération d'ajout d'un élève (donné avec toutes ses informations) par insertion séquentielle ordonnée, afin de maintenir l'ordre des élèves par numéro de NIP et profiter de l'accès efficace précédent.

1. On supposera disposer d'une relation d'ordre alphanumérique sur les chaînes de caractères, applicable aux NIP, notée $<$, $=$, $>$. Cette relation d'ordre est en général fournie par les langages, notamment en C : fonction `strcmp` ("string compare") de la bibliothèque `<string.h>` de manipulation de chaînes.

TD Structures de données

Polytech Lille, IMA3

Objectifs Algorithmes de base sur les listes chaînées et traduction en C.

1 Algos

On considère des listes d'entiers comme vues en cours. Pour chacun des algorithmes (en pseudo-code, itératif), on donnera une évaluation de coût.

1. Concevoir un algorithme de concaténation de deux listes chaînées.
2. En reprenant les principes vus en cours, concevoir l'algorithme complet d'insertion d'un élément (entier) dans une liste chaînée donnée triée, en prévoyant tous les cas.
3. En reprenant les principes vus en cours, concevoir l'algorithme complet de suppression d'un élément `elem` entier donné dans une liste chaînée, en prévoyant tous les cas.

2 Listes chaînées en C

- Ecrire en C les fonctions de ce ce dernier algorithme
- Montrer son utilisation dans un main (déclaration d'une liste et appel à la suppression d'un élément).
- Faire un schéma d'exécution montrant le passage de paramètres.

TD de structures de données
Récursivité, listes et arbres

Polytech'Lille IMA S6 2011-2012

1 Algorithme récursif de calcul de x^n.

2 Ecrire un algorithme qui calcule le nombre d'occurences d'un élément dans une liste.

3 Ecrire un algorithme qui teste si une liste est un ensemble (pas de doublons).

4 Ecrire un algorithme récursif de suppression d'un élément dans une liste (on supprimera la première occurrence).

5 Et si la liste est ordonnée croissante ?

6 Ecrire un algorithme qui affiche le feuillage (ensemble des feuilles) de gauche à droite d'un arbre binaire.

7 Ecrire un algorithme qui calcule la hauteur d'un arbre binaire.

TD de structures de données
Structures, Tables ordonnées, indirection

Polytech'Lille IMA S6 2011-2012

On s'intéresse à quelques opérations de gestion d'une table d'ouvrages d'une bibliothèque. Les ouvrages sont décrits par les informations suivantes : titre, auteur, numéro de code (ISBN), année de parution, localisation en bibliothèque (+ d'autres informations éventuelles telles qu'un résumé, des mots clés, ...).

La bibliothèque n'est pas figée et on doit pouvoir ajouter de nouveaux ouvrages par une opération `ajout`. Ces mises à jour sont cependant secondaires (elles se font en dehors des heures d'ouverture de la bibliothèque) par rapport aux opérations de recherche interactive d'ouvrage par les usagers.

1 Accès rapide par code

On doit pouvoir rechercher rapidement un ouvrage par son code (ISBN). Concevoir une première SD en reprenant les principes vus au 1er semestre.

Remarque : il n'est pas demander d'écrire d'algorithmes.

2 Généralisation : accès multiples

On peut multiplier les critères d'accès à la base d'ouvrages, comme par exemple :

- par code localisation
- par titre

Ces opérations de recherche sont privilégiées, de la même façon que la recherche par code ISBN, par rapport aux opérations de mise à jour.

- Analyser ce problème pour trouver une nouvelle structure de données qui place au même niveau tous les critères d'accès sur la base d'ouvrages.
- Ecrire les algorithmes de recherche (par exemple par code) et d'ajout d'un ouvrage.

TD de structures de données
Files d'attente

Polytech'Lille IMA S6

Le service informatique d'une chaîne de grands magasins désire fournir à ses succursales un logiciel de simulation du comportement de leurs clients à l'arrivée aux caisses.

Il existe plusieurs tailles de magasins dont le nombre de caisses peut varier entre 20 et 30.

Certains clients sont privilégiés et bénéficient d'une priorité sur les autres à certaines caisses dites "spéciales".

Pendant l'ouverture du magasin, certaines caisses sont fermées.

Les règles d'utilisation des caisses sont :

- toute caisse, spéciale ou non, peut être empruntée par tout client, privilégié ou non.
- les clients privilégiés sont prioritaires sur les autres aux caisses spéciales.
- dans une même catégorie (privilégiés/autres) un client ne se range jamais devant un autre dans la file d'attente (!)
- pour simplifier on considère constant le temps de passage d'un client à une caisse quand c'est son tour. Seul le nombre de clients dans une file intervient donc dans l'estimation du temps d'attente.
- un client choisit systématiquement la "meilleure caisse", définie comme celle dont la file offre le temps d'attente minimum. Un client privilégié exploite toujours son droit de priorité aux caisses spéciales. Un client non privilégié préfère toujours aller à une caisse ordinaire quand l'occupation est la même qu'une caisse spéciale.
- on considère que les clients arrivent un par un aux caisses, c'est à dire que l'occupation ne varie pas pendant qu'un client choisit une caisse.

On associe à chaque client le coût de ses courses et à chaque caisse son chiffre d'affaire (cumul des coûts des courses des clients qui sont passés à cette caisse).

On s'intéresse aux opérations suivantes :

- `comportementStandard` qui simule le comportement d'un client non privilégié,
- `comportementPrivilegie` qui simule le comportement d'un client privilégié,
- `auSuivant` qui, pour une caisse donnée, simule le passage du client dont c'est le tour.

Proposez une structure de données pour résoudre ce problème et écrire les algorithmes des opérations précédentes.

POLYTECH'Lille
Département IMA
3ème année

7 Juin 2010

Devoir surveillé de Structures de données
(Tous documents papiers autorisés)
Durée : 2 heures
Il est demandé de formuler vos solutions en pseudo-langage et non en langage C

On s'intéresse dans ce problème aux circuits touristiques d'une région donnée, au hasard la région Nord. Les structures de données utilisées dans le problème sont les suivantes :
Les N villes touristiques de la région sont répertoriées dans un vecteur de taille N. On définit le type correspondant de la façon suivante :

type Villes = Vecteur[N] de chaîne-de-caractères

Exemple : Soit v : Villes

v	Cambrai	Douai	Hautmont	Hazebrouck	Lille	Maubeuge	Roubaix	Tourcoing	Valenciennes
	0	1	2	3	4	5	6	7	8

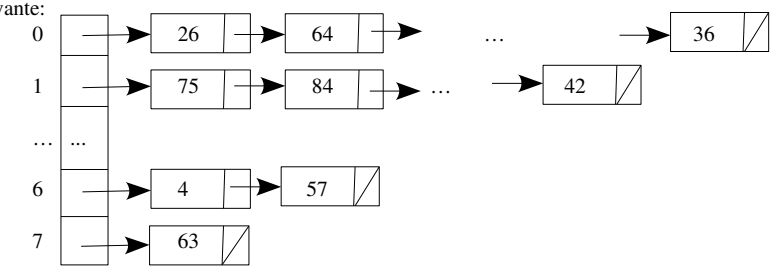
On souhaite étudier deux implémentations pour représenter les distances entre les villes :
- La première consiste à représenter directement la matrice par un tableau "distance" de taille N tel que distance[i][j]=distance[j][i]=distance entre les villes i et j.

Exemple : Soit distance, le tableau suivant :

	0	1	2	3	4	5	6	7	8
0	0	26	64	115	74	70	82	88	36
1	26	0	75	84	45	76	52	56	42
2	64	75	0	127	87	6	91	97	35
3	115	84	127	0	43	128	55	56	92
4	74	45	87	43	0	88	11	15	52
5	70	76	6	128	88	0	92	98	36
6	82	52	91	55	11	92	0	4	57
7	88	56	97	56	15	98	4	0	63
8	36	42	35	92	52	36	57	63	0

- La distance entre Cambrai et Douai est de 26 km
- La distance entre Valenciennes et Douai est de 42 km

- La seconde part du constat que cette matrice est symétrique. On envisage alors la représentation suivante:



A l'entrée i de la table est associée la liste des distances entre la ville v[i] et les villes v[i+1], v[i+2], …, v[N-1].

Question 1 : Sachant qu'un entier et qu'un pointeur occupent la même taille mémoire, soit X octets, comparer les coûts mémoires de ces deux solutions.

Question 2.1 : Déclarer la SD correspondant à la première solution et écrire la fonction qui détermine la distance entre deux villes données par leur nom.

Question 2.2 Quel est le coût de cette fonction sur cette solution?

Question 3.1 : Déclarer la SD correspondant à la deuxième solution et écrire la fonction qui détermine la distance entre deux villes données par leur nom.

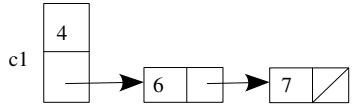
Question 3.2 : Quel est le coût de cette fonction sur cette solution?

Dans la suite du problème, on retient la première solution
Un circuit touristique est décrit par la ville de départ/arrivée et la liste des villes étapes (le circuit part d'une ville, passe par des villes étapes et revient à cette ville). L'ensemble des villes visitées comprend donc la ville départ/arrivée plus les villes étapes. La ville de départ/arrivée et les villes étapes sont désignées par l'indice où elles sont rangées dans le vecteur villes. Pour l'exemple, l'entier 4 désigne Lille (v[4]).
Un circuit est donc représenté par la structure de données suivante :

```
type PtCellule =  Pointeur de Cellule
type Cellule = Structure {
        etap : Entier
        etapSuiv : PtCellule
}

type Circuit = Structure {
        depart : entier
        listeEtap : PtCellule
}
```

Exemple : Soit c1 de type Circuit

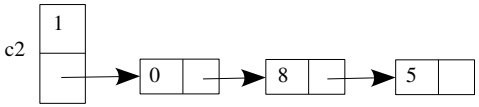


c1 représente le circuit dont la ville de départ/arrivée est Lille (v[4]). Les villes étapes sont successivement Roubaix et Tourcoing. Les villes visitées sont donc Lille, Roubaix, Tourcoing.

Question 4 : Ecrire un algorithme qui, étant donnés un circuit c de type Circuit et un tableau distance calcule la distance totale à parcourir pour effectuer le circuit complet.
Pour l'exemple du circuit c1 représentant Lille-Roubaix-Tourcoing-Lille, la distance totale de c1 est de 30 km (11 km de Lille à Roubaix et 4 km de Roubaix à Tourcoing, 15 km de Tourcoing à Lille).

Question 5 : Ecrire un algorithme qui pour un circuit c de type Circuit, un tableau distance et une étape e donnés détermine le nombre de kilomètres économisés si on ne passe pas l'étape e (l'étape e est donnée par un entier tel que v[e] désigne la ville étape à éviter). On supposera ici que le circuit c passe une seule fois par l'étape e.

Exemples :
Pour le circuit c1 et e=7, on gagne 8 km. En effet, il y a 4 km de Roubaix à Tourcoing et 15 km de Tourcoing à Lille soit 19 km pour faire Roubaix-Tourcoing-Lille contre 11 km pour faire Roubaix-Lille sans passer par Tourcoing.
Pour le circuit c2 suivant et e=0, on gagne 20 km.



En effet, il y a 26 km de Douai à Cambrai et 36 kilomètres de Cambrai à Valenciennes, soit 62 km pour faire Douai-Cambrai-Valenciennes contre 42 km pour faire Douai-Valenciennes sans passer par Cambrai.

Question 6 : Ecrire un algorithme qui détermine si un circuit c de type Circuit donné passe au moins 2 fois par la même ville.
Exemples :
On ne passe pas deux fois dans la même ville dans les circuits « Lille-Roubaix-Lille » ou « Douai-Cambrai-Valenciennes-Douai ». En revanche, on passe deux fois dans la même ville dans les circuits « Douai-Cambrai-Valenciennes-Cambrai-Douai » ou « Lille-Roubaix-Lille-Tourcoing-Lille ».