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.

IMA 3 - T.D. - Programmation Avancée
Piles, Files

1 Ordonnancement par file de priorités

Tout processus a besoin d'accéder au CPU pour pouvoir s'exécuter. *L'ordonnanceur* est l'élément du Système d'Exploitation responsable du choix de l'ordre d'exécution des processus sur le (ou les) CPU. Dans cette section, on s'intéresse à une structure de données permettant d'implémenter une politique d'ordonnancement basée sur les priorités sous Unix.

La politique que nous considérons combine deux mécanismes :

1. **Priorité** : l'ordonnanceur choisit toujours d'exécuter le (un des) processus de priorité maximale qui est en attente. Les processus de même priorité sont exécutés dans leur ordre d'arrivée ;
2. **Tourniquet (round robin)** : lorsqu'un processus s'exécute, il s'exécute au maximum pour un quantum de temps fixé, disons 5ms. A l'issue du quantum alloué, si le processus n'est pas terminé il est préempté, remis en attente, et l'ordonnanceur passe au processus suivant.

Un processus est caractérisé par :
— Son identifiant entier dit *process id* (pid) ;
— Son nom ;
— L'identifiant de l'utilisateur propriétaire ;
— Une priorité d'exécution allant de 0 (plus prioritaire) à 40 (moins prioritaire).

1. Écrivez la structure processus qui représente un processus Unix ;
2. Proposez une (ou plusieurs) structure de données `file_prio` pour gérer la file de priorité de l'ordonnanceur. Comparez différentes solutions possibles ;
3. Écrivez une fonction `ajouter` paramétrée par une `file_prio` et un `processus`, qui ajoute le processus dans la file de priorité ;
4. Écrivez une fonction `suivant`, paramétrée par une `file_prio`, qui renvoie (par valeur de retour ou paramètre) le processus suivant à exécuter.

2 Expressions arithmétiques post-fixées

Vous avez l'habitude de manipuler des expressions arithmétiques sous leur forme *infixe*, c'est-à-dire que les opérateurs apparaissent entre leurs opérandes. Il existe aussi une notation *préfixe* (opérateurs avant leurs opérandes) et *postfixe*

(opérateurs après leurs opérandes). Par exemple, les 3 expressions suivantes produisent le même résultat :

$$\begin{aligned} 2 + 3 * 4 \quad & \text{(infixe)} \\ + \ 2 \ * \ 3 \ 4 \quad & \text{(préfixe)} \\ 2 \ 3 \ 4 \ * \ + \quad & \text{(postfixe)} \end{aligned}$$

Dans la suite, nous allons nous concentrer sur la notation post-fixée.

1. Utilisez une pile pour calculer l'expression suivante : $3\ 4\ 2\ *\ +\ 1\ -$
2. Définissez un type `lexeme` pour représenter les différents composants d'une expression arithmétique. On incluera un lexeme `End` désignant la fin d'une expression ;
3. On suppose l'existence de la fonction suivante qui renvoie le prochain lexeme de l'expression en cours d'analyse : `lexeme lexSuivant()`. Ecrire une fonction permettant de calculer le résultat d'une expressions arithmétique postfixée ;
4. Convertissez l'expression suivante en son équivalent postfixé : $1*5+4-2$
5. Ecrire une fonction permettant d'afficher l'expression postfixée correspondant à une expression infixée analysée à l'aide de la fonction `lexSuivant()`.

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.