POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S6) / 2013 **Programmation Avancée**

Jonathan Aceituno, Vincent Aranéga, Nathalie Devésa, Rachid Elkouri et Laure Gonnord

TP4 Listes chaînées V1 : algorithmes impératifs

Objectifs

- Savoir déclarer, construire des listes chaînées en C.
- Savoir reconnaître et implémenter les algorithmes classiques de listes chaînées à l'aide de fonctions *impératives*.

Contexte et préparation Nous allons utiliser les listes chaînées (non ordonnées, puis ordonnées) pour représenter des **ensembles d'entiers**. Les opérations classiques sur les ensembles d'entiers seront réalisées à l'aide d'algorithmes classiques sur les listes, en version **impérative**.

1 Questions du TP (à faire impérativement)

1.1 En direct du cours - Listes non triées

Les fonctions suivantes ont été traitées en TD.

1. Déclarer un type cellule, pointeur de cellule, liste d'entiers.
2. Écrire une fonction d'ajout en tête d'un entier dans une liste d'entiers (avec allocation).
3. Écrire une fonction d'impression d'une liste d'entiers.
4. Écrire une fonction de suppression de la première cellule d'une liste, puis une fonction de désallocation d'une liste.
5. Tester les fonctions précédentes (déclaration d'une liste vide dans le `main`, appels des fonctions précédentes. On lancera `valgrind` pour vérifier l'absence de fuite mémoire à l'exécution.

1.2 Listes triées

On va maintenant faire en sorte de maintenir une liste triée **sans doublon**.

1. Écrire un algorithme d'insertion d'un entier dans une liste supposée triée par ordre croissant supposée coder un ensemble. On ne fera donc une insertion que si l'élément n'existe pas. La fonction aura la signature

   ```
   void insertInSortedList (List* p_list, int el, bool* res);
   ```

   la variable res valant `true` en cas de véritable insertion, `false` si l'élément existait.

   Comme dans le cours, on commencera par écrire une fonction qui recherche l'adresse de la cellule qui précédera la nouvelle cellule à insérer.

2. Écrire une fonction qui permet à l'utilisateur de créer une liste d'entiers par lecture dans un fichier. On supposera que le fichier contient un entier par ligne. Les entiers pourront être négatifs, et dans n'importe quel ordre (utiliser la fonction précédente). **On pourra évidemment s'inspirer du TP2 pour la lecture dans un fichier**. On testera avec un fichier ouvert dans le `main`.

3. Écrire une fonction qui retourne la taille d'une liste passée en paramètre (c'est à dire ici le cardinal de l'ensemble).

4. Écrire une fonction qui permet de tester si une liste quelconque d'entiers est triée.

2 Questions s'il vous reste du temps

Écrire des fonctions/procédures pour

1. tester si une liste quelconque d'entiers n'a pas de doublons

2. faire l'union de deux ensembles codés sous forme de liste triées d'entiers (puis l'intersection, ...)

3. transformer une liste quelconque en un ensemble (liste triée sans doublon)

On se posera à chaque fois la question de la complexité des fonctions écrites.

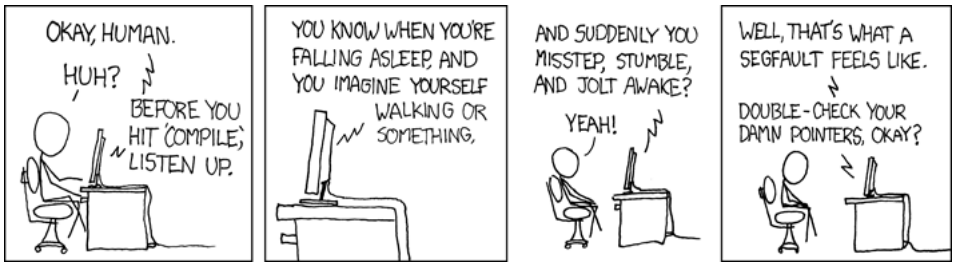

Figure 1: Compiler Complaint, http://xkcd.com/371/, sous License Creative Commons