POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2012/2013**

http://laure.gonnord.org/pro/teaching/

TP5 - Matrices et redirections + simulation de TP noté

Objectifs

- Manipuler les entrées/sorties C (via les redirections).
- Implémenter en C quelques algorithmes simples de matrices
- Découvrir et implémenter l'algorithme classique de recherche dichotomique dans un tableau trié.
- Expérimenter le système de rendu de TP noté.

On commencera par créer un répertoire TP5 dans le répertoire Algo

1 Fonctions simples sur les matrices

Avant de commencer, copier le fichier `alea.c` du répertoire `TP4` dans le répertoire courant (TP5) , puis le renommer en `tp5.c`. Éditer ce fichier, puis écrire un programme principal (main), puis :

1. Une procédure d'initialisation d'une matrice d'entiers avec des entiers tirés au hasard (vous pouvez faire des copier-coller à l'aide du tp précédent).
2. Une procédure d'impression d'une matrice d'entiers, de taille `nbl*nbc`, de signature :

   ```
   void imprimeMat(int mat[NBL][NBC]);
   ```

   Combien votre algorithme fait-il d'impressions ? (à noter en commentaire).
3. Déclarer une matrice de taille 10*12 dans le main et tester les deux procédures précédentes.

On remarquera qu'ici on utilise les constantes symboliques pour donner les tailles des matrices. On fait cela car le compilateur doit obligatoirement connaître la taille d'une ligne de la matrice avant compilation.

2 Simulation de rendu de TP noté

Dans cette section, vous allez utiliser Moodle pour rendre votre exercice précédent. Pour pouvoir faire cela il faut obligatoirement avoir fait les manipulations décrites dans le mail envoyé pour PS (inscription au cours, dans le bon groupe).

1. Renommer votre `tp5.c` en `tp5_monlogin.c`, avec `monlogin` remplacé par votre login à polytech. Si vous êtes en binôme, faire deux copies (`tp5_login1.c`, `tp5_login2.c`).
2. Ouvrir un navigateur à l'adresse https://moodle.polytech-lille.fr/login/index.php et se loguer (login, mot de passe polytech).
3. Aller dans le cours "Programmation Structurée" (colonne de gauche), puis sur "test d'un dépôt de fichier", et dans le bon groupe de TP. Uploader le fichier `tp5_monlogin.c` et ajouter un commentaire "ceci est le tp5 de xxxx" (nom et prénom).
4. Se déloguer.

3 Matrices et entrées/sorties par redirection

Récupérer des données stockées dans un fichier dans une matrice C

1. Récupérer les fichiers `getmat.c` et `thegrid.txt` sur la page du cours, avec `wget`, dans le répertoire courant.
2. Compiler le fichier `getmat.c` en `getmat` (clang)
3. Ce binaire attend des informations sur le terminal (utilisation de *fgets* mais on aurait pu utiliser *scanf*). Au lieu de fournir ces informations, on va utiliser les informations d'un fichier (ici `thegrid.txt`) "comme si" elles étaient écrites sur le terminal (autrement appelé "entrée standard"), en faisant :

   ```
   ./getmat <thegrid.txt
   ```

Des informations plus détaillées sur les redirections peuvent être trouvées par exemple à l'adresse http://www.tuteurs.ens.fr/unix/shell/entreesortie.html.

Deux fonctions simples Dans le fichier `getmat.c` à éditer comme d'habitude avec *emacs* :

1. Écrire une fonction qui compte le nombre de fois qu'un caractère donné apparaît dans la matrice. Combien de tests fait votre fonction ? (mettre en commentaire dans votre programme).
2. Écrire une procédure qui construit la matrice symétrique de la matrice d'entrée. On écrira une procédure de signature :

   ```
   void construitSym(char mat[N][N], char resu[N][N])
   ```

   mat étant la matrice donnée, et *resu* la matrice résultat.
3. Tester (sur la matrice fournie) en imprimant les résultats (vous pouvez copier-coller la fonction d'impression précedemment écrite).
4. Tester la redirection de la sortie standard :

   ```
   ./getmat <thegrid.txt >resultats.txt
   ```

   Observer que plus rien n'est écrit sur le terminal, et observer le contenu du fichier `resultats.txt`

4 Recherche dichotomique

Dans cette section, on va implémenter deux algorithmes de recherche dans un tableau et comparer leurs temps d'exécution. Comme dans les deux sections précédentes, tous les tableaux de cette section auront une taille `N` constante symbolique déclarée en haut du programme (`#define N 10`)

1. Écrire une fonction qui permet de récupérer sur l'entrée standard un tableau d'entiers de taille N constante fixée à l'avance (on suppose que les entiers entrés sont séparés par "entrée"). On utilisera `scanf("%d")` et une boucle pour. Tester.
2. Écrire une fonction `bool recherche1(int tab[N], int el)` qui recherche l'élément `el` dans le tableau `tab` en parcourant le tableau de gauche à droite. Combien de tests effectue au pire votre fonction ?

3. Écrire une fonction `bool recherche2(int tab[N], int el)` qui recherche dans le tableau `tab` supposé *trié*. On utilisera un algorithme de recherche dichotomique [1] dans un tableau, dont le principe est le suivant :
   - Regarder la case du milieu du tableau. Si $t[mil] = el$ c'est trouvé
   - Sinon, si $t[mil] > el$, comme le tableau est trié on sait que l'élément, si il existe est dans la partie gauche du tableau.
   - Sinon, cela signifie que $t[mil] < el$ et il faut chercher dans la partie droite.

   Combien de tests effectue au pire votre fonction ?
4. Avant de passer à la suite, tester vos fonctions pour N=10. On pourra initialiser les tableaux comme ceci :

   ```
   int tab[N] = {0, 2, 3, 78, 80, 200, 201, 1515, 2048, 10000};
   ```

   au lieu de lire sur l'entrée standard.
5. On fournit sur la page du cours un fichier comportant N éléments dans l'ordre croissant. Comparer les temps d'exécution de vos deux fonctions de recherche sur ce tableau de taille N=10000. Si on ne voit pas de différence, on imprimera quand même le nombre d'étapes de chacun des deux algos (ajout d'un compteur de nombre de tests).

5 Si vous avez encore du temps

Traiter le problème : http://projecteuler.net/problem=18

1. http://fr.wikipedia.org/wiki/Dichotomie