POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2010/2011**

TP5 et 6 - Trions !

Objectifs

Coder et comparer les différents tris vus en cours. Début de la compilation séparée. Visualisation graphique. Utilisation de gnuplot.

IMPORTANT : à chaque fin de section, faites vérifier votre travail par un enseignant !

Unix - Wget / Tar

1. Récupérez sur la page web du cours l'archive `tp5.tgz` (copiez l'adresse à partir d'un navigateur, puis utilisez wget).
2. Placez dans *votre répertoire de travail pour la programmation structurée* (commande mv).
3. Désarchivez avec la commande :

   ```
   tar xvf tp5.tgz
   ```

   (x pour extraire, v pour verbose, f pour file). La création du répertoire `TP5_codes` est normalement automatique.

1 Préparation : compilation de plusieurs fichiers

Au tp4, nous avons créé un fichier contenant des fonctions de base sur les vecteurs d'entiers (initialisation avec des valeurs aléatoires, impression sur le terminal, ...).

L'archive fournit une correction des fonctions les plus importantes. On fournit le fichier `vecteurs.c` qui contient ces fonctions, et le fichier `vecteurs.h` qui fournit les déclarations de ces fonctions.

1. Observer les fichiers `vecteurs.c` et `vecteurs.h`. Remarquer que le .c ne contient pas de fonction principale (*main*), et que le .h ne contient pas de code des fonctions, juste leurs signatures.
2. Compiler `vecteurs.c` avec la commande :

   ```
   gcc -c vecteurs.c
   ```

   Cette commande crée un fichier `vecteurs.o` (vérifier), qui est un fichier objet (voir la feuille annexe sur la compilation)
3. Créer le fichier `tris.c` (avec xemacs), inclure le fichier d'entête avec `#include "vecteurs.h"`, puis dans le programme principal faites des appels aux fonctions d'initialisation et d'impression.

4. Compiler avec

   ```
   gcc -c tris.c
   gcc -o tris vecteurs.o vecteurs.o
   ```

5. Exécuter. Tester.
6. À ce stade, faites une archive compressée avec les fichiers .c et .h, et pas le reste :

   ```
   tar cvzf tp5_amoi.tgz vecteurs.h vecteurs.c tris.c
   ```

   puis vérifiez (dans un autre répertoire), que vous savez le décompresser.

2 Tris non graphiques

Dans le fichier `tris.c`, écrire et tester les tris du cours suivants :

1. Tri insertion.
2. Tri sélection.
3. Tri fusion.

On écrira une procédure par tri, et on testera dans le main, sur des tableaux de taille 10 initialisés au hasard.

3 Visualisation graphique de l'exécution de tris

3.1 Installation de la librairie graphique

On va utiliser la librairie graphique se trouvant à l'adresse : http://para.inria.fr/~peskine/enseignement/iup-2002/graphics-doc.html (lien dispo à partir de la page web du cours).

1. Télécharger la librairie, et trouver un bon endroit pour l'installer.
2. Extraire l'archive (même commande). Un répertoire `Libgraphics/` est créé.
3. Aller dans ce répertoire et tapez `make` (compilation de la librairie).
4. Vérifier que tout est bon en tapant `make test`, et en exécutant `./test`.
5. Noter sur un papier le chemin qui mène à `Libgraphics/` : pour l'avoir l'avoir, lancer (par exemple) la commande `path` lorsque vous êtes à l'intérieur du répertoire `Libgraphics/`

3.2 Test de la librairie graphique

Dans le fichier `testgraphic.c`, certaines fonctions de la librairie graphique ont été appelées. Pour pouvoir compiler et exécuter il faut utiliser des commandes qui *lient* la bibliotheque au code c que l'on écrit. Ces commandes sont écrites dans le fichier `Makefile` qui vous est fourni dans l'archive.

1. Ouvrir avec xemacs ce Makefile.
2. Modifier le chemin dans les lignes de configuration en haut : la variable `GRAPHICSLIB` doit contenir le chemin vers le fichier `libgraphics.a` et `GRAPHICSH` doit contenir le chemin vers `graphics.h` [1].

1. la justification de tout cela est dans la feuille annexe de compilation, et sera vue plus tard également

3. La commande `make test` vous permet de compiler un exécutable. Observer le fichier `testgraphic.c`. qui est le fichier compilé lors de la commande `make test`.

3.3 Tri graphique

- Copier votre fichier `tris.c` en `trisg.c`.
- Vérifier que le fichier `trisg.c` compile toujours en tapant `make` (la liaison avec la lib graphique est faite dans le Makefile).
- Dans le fichier `trisg.c`, écrire une fonction `triInsertionGraphique` qui réalise le tri insertion d'un tableau et qui l'illustre sur la fenêtre graphique.

  On pourra utiliser une fenêtre graphique de taille 500*500 et la macro suivante :
  `#define RECT(j) (gr_fill_rect(13+10*(j),t[(j)],20+10*(j),10))`
  qui dessine un rectangle proportionnel au nombre contenu à la case $t[j]$ (t, pas un autre nom !)

4 Données expérimentales sous forme graphique : gnuplot

4.1 Prise en main de Gnuplot

1. Observer les fichier `gnuplot.gp` et `donnees.dat` (de l'archive) et utiliser gnuplot pour imprimer la courbe de points dans un fichier ps :

   ```
   gnuplot gnuplot.gp
   ```

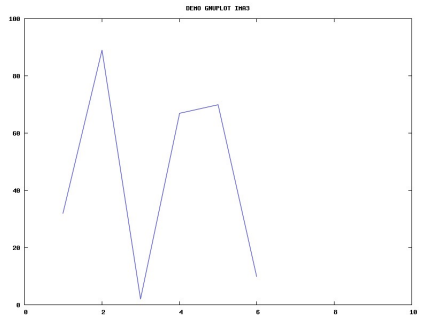
   (on pourra utiliser gv ou evince pour lire le fichier postcript)

2. Créer un programme C qui demande 6 entiers positifs à l'utilisateur, puis les imprime sur la sortie standard (printf) de la façon suivante :

   ```
   1 entier1
   2 entier2
   ...
   ```

   Compiler, tester.

3. Rediriger la sortie standard pour récupérer les informations entrées par l'utilisateur, puis créer un jpg avec ces données (cf documentation web de gnuplot) :



4.2 Utilisation avec les données expérimentales

Faire des jolis graphiques avec les temps d'exécution de vos tris (commande time).