Licence **UE INF 255**
Université Bordeaux 1 **Projet de Programmation 2**

Présentation

Le but de ce projet est d'implémenter la structure de données appelée *tas* qui est en fait un arbre binaire « quasi complet » (ou encore « quasi parfait ») respectant une relation d'ordre. Rappelons qu'un arbre binaire quasi-complet est un arbre dont tout les étages sont remplis sauf éventuellement le dernier qui doit être tassé vers la gauche. Une façon efficace d'implémenter un tel arbre est de numéroter ses sommets selon un parcours en largeur et d'utiliser cette numérotation pour faire correspondre les sommets aux éléments d'un tableau. Une numérotation classique est de numéroter 1 la racine et pour tout sommet numéroté i de numéroter ses fils gauche et droit $2i$ et $2i + 1$. Dans un tas, la relation d'ordre est utilisée pour faire en sorte que la valeur associée au père soit toujours supérieure à celle de ses fils. Implémentées efficacement, l'insertion d'un élément quelconque dans un tas et l'extraction du tas de l'élément maximal (toujours associé à la racine) sont deux opérations qui peuvent se faire en le logarithme de la taille du tas tout en conservant la structure de tas. Ainsi, un tri basé sur l'utilisation d'un tas a une complexité en $O(n \log(n))$. De tels algorithmes sont écrits de façon précises dans les cours d'algorithmes et structures de données suivants :

- http://www.labri.fr/perso/maylis/ASDF
- http://dept-info.labri.u-bordeaux.fr/~cori/Enseirb/poly.pdf
- http://graal.ens-lyon.fr/~yrobert/algo/poly-algo.ps

Objectifs

Pour ce projet, il s'agit d'écrire un module *tas* générique permettant de stocker des structures contenant des entiers, des flottants, des chaînes de 24 caractères... De façon générale la seule contrainte est que pour un tas donné toutes les informations aient la même taille. Les fonctionnalités sont les suivantes :

- créer et libérer un tas associé à une relation d'ordre ;
- insérer dans un tas un élément ;
- extraire l'élément maximal d'un tas ;
- donner le nombre d'éléments contenu dans un tas ;
- charger / modifier / stocker un tas contenu dans un fichier.

L'utilisation de fichiers permettra la manipulation d'un tas par différents processus à des moments différents. Notons qu'il n'est pas judicieux de stocker des pointeurs dans un fichier. Dans ce projet, on copiera directement dans le tas les informations à conserver. On pourra cependant créer des tas de pointeurs mais il ne faudra pas chercher à sauvegarder de tels tas.

Pour programmer ce module nous vous proposons d'étudier soit l'approche dynamique, soit l'approche statique :

Approche dynamique. Il s'agit de s'inspirer de la technique de programmation de la fonction de tri générique `qsort()`. Pour travailler, cette fonction a besoin du nombre d'éléments contenus dans le tableau à trier, de la taille unitaire des éléments et d'un pointeur de fonction. Pour créer un tas générique, il faut préciser de façon analogue la taille des éléments à stocker et la

fonction de comparaison à utiliser pour les ordonner. On aura intérêt à définir une fonction ou macro `sommet(tas, numéro)` qui retournera l'adresse du sommet en question et à utiliser la fonction `memcpy()` pour recopier les données.

Approche statique. L'objectif est de générer le code spécifique à chaque type à partir d'un modèle générique (gabarit ou *template*, en anglais) à l'aide d'un préprocesseur. Cette technique de programmation fut utilisée aux origines du langage C++. L'intérêt de cette technique est double : on dispose d'un code optimisé pour chaque type et le code source (celui écrit par le programmeur) n'est pas dupliqué car il écrit un méta-code à partir duquel sont générés les codes spécifiques. On trouvera sur le site d'Achille Braquelaire http://dept-info.labri.fr/~achille/enseignement/INF-153/src/Chapitre-09 un code source mettant en oeuvre cette technique (on pourra comparer le méta-code au code présenté au Chapitre-06). Dans notre cas, il s'agit de paramétrer le module par le type de données à stocker et la fonction de comparaison.

Pour illustrer le bon fonctionnement du module on écrira deux applications :

- manipulation de tas d'entiers stockés dans des fichiers : on définira un ensemble de programmes exécutables pour manipuler un tas donné en paramètre (création, destruction, insertion, extraction du sommet) ;
- tri de chaînes de caractères : on écrira une commande `trier` qui affiche de façon ordonnée les lignes présentes sur son entrée. Le tri pourra s'effectuer suivant l'ordre lexicographique (par défaut) ou l'ordre numérique ascendant mais pourra aussi se faire suivant l'ordre inverse ou de façon descendante. L'usage de ce programme sera le suivant :

```
usage : trier [-i] [-n]

-i : inverser l'ordre
-n : choisir l'ordre numérique
```

Quelques précisions techniques

On appliquera le principe du masquage de l'implémentation tout en soignant l'efficacité de l'implémentation proposée, en particulier, on évitera l'utilisation de procédures récursives.

À l'aide du module `projection` (http://dept-info.labri.fr/ENSEIGNEMENT/projet2/supports/Tas) basé sur l'appel système `mmap()`, on projettera le fichier du tas en mémoire. C'est une méthode très efficace pour lire et écrire dans un fichier aussi simplement que de manipuler un tableau. À cet effet, un code vous sera donné pour illustrer le procédé de projection du fichier de tas.

Échéances

Première étape : il s'agit d'avoir programmé le tas pour le type entier (sans s'occuper de la partie généricité ni de l'interface) et d'avoir essayé le module `projection` pour sauvegarder un fichier. Il faudra avoir préparé ses questions sur la partie générique et avoir une idée de l'interface générique.

Seconde étape : il s'agit de généraliser le travail à la généricité.