TP1 Algorithmes récursifs - Tableaux

AL31/CNAM STMN A1, J.-M. ROBERT

30 septembre 2019

Remarque préliminaire :
Chacune des fonctions implantées sera testée depuis une fonction principale (`int main()`). Le travail effectué sera déposé sur le Moodle sous la forme d'une fichier source. Les réponses aux questions posées seront insérées en commentaire dans le code source.

1 Algorithmes récursifs

1.1 Suites définies par récurrence

1. Implanter une fonction dont le prototype est le suivant (implantation itérative) :
   `unsigned long int factorial(int n)`, retournant $n!$.
2. Implanter une fonction récursive dont le prototype est le suivant :
   `unsigned long int rec_factorial(int n)`, retournant $n!$.

   On définit la suite de Fibonacci de la façon suivante :

$$u_1 = 1, u_2 = 1,$$
$$u_n = u_{n-1} + u_{n-2}, \quad \forall \quad i > 2.$$

3. Implanter une fonction dont le prototype est le suivant (implantation itérative) :
   `unsigned long int fibonacci(int n)`, retournant u_n (le $n^{\text{ème}}$ terme de la suite de Fibonacci).
4. Implanter une fonction récursive dont le prototype est le suivant :
   `unsigned long int rec_fibonacci(int n)`, retournant u_n.
5. Pour chacune de ces implantations, évaluer la complexité en fonction du paramètre d'entrée `n`. Que concluez-vous de ces complexités ?

   Remarque 1. Vous pourrez vérifier que le rapport entre deux éléments consécutifs de la suite de Fibonacci (u_n et u_{n-1}) tend vers le nombre d'or, soit :

$$\varphi = \frac{1+\sqrt{5}}{2}$$

2 Tableaux

Le but de ce travail est d'implanter et de tester différents algorithmes de traitement sur les tableaux.

L'implantation des algorithmes se fera en C dans Visual Studio (on pourra utiliser les fichiers sources du Moodle). Chacune des fonctions implantées sera testée depuis une fonction principale (`int main()`).

2.1 Allocations mémoire

1. Déclarer et allouez quatre tableaux de taille `SIZE` éléments de type `int` (voir le `#define`) :
   — un tableau `t0` en mémoire globale avec allocation statique ;
   — un tableau `t1` en mémoire locale (dans la fonction `int main()`) avec allocation statique ;
   — un tableau `t2` en mémoire locale (dans la fonction `int main()`) avec allocation dynamique (utilisez la fonction `malloc()`) ;
   — un tableau `t3` en mémoire locale (dans la fonction `int main()`) préfixé avec le mot clé `static` et avec allocation statique ;
2. Afficher les valeurs des pointeurs (format `%p`) et vérifiez avec le debugger de Visual Studio. Que constatez vous ?

2.2 Implantation

pour chaque fonction implantée, il vous faut la tester en l'appelant à partir de la fonction principale `main` et en affichant le résultant (en utilisant la première procédure implantée).

1. Implanter une procédure d'affichage d'un tableau de `size` entiers dont le prototype est : `void affiche_tableau(int *tab, int size)` ;
2. Implanter une fonction d'initialisation d'un tableau de `size` entiers (déjà alloué en mémoire), affectant à chaque élément une valeur aléatoire entre `0` et `MAX_VAL-1`, macro définie en en-tête du fichier source (on utilisera la fonction `rand()` qui renvoie un entier entre 0 et 2^{31}) dont le prototype est :
   `int init_tableau(int * tab, int size)` ;
3. Implanter les fonctions de recherche dans le tableau précédent de l'élément minimum et l'élément maximum. Les prototypes sont les suivants :
   — `int min_tab(int * tab, int size)` ;
   — `int max_tab(int * tab, int size)` ;
   Ces deux fonctions renvoient l'indice de l'élément correspondant dans le tableau.
4. Implanter une procédure d'échange de deux éléments d'un tableau dont le prototype est le suivant (`elt1` et `elt2` sont les indices des éléments à échanger) :
   `void swap(int * tab, int elt1, int elt2)` ;
5. Implanter une procédure de tri sélection dont le prototype est le suivant :
   `void selection_sort(int *tab, int size)`. Cette procédure modifie le tableau passé par son pointeur, et fera usage des fonctions et procédures précédentes.
6. Implanter une fonction de tri rapide dont le prototype est le suivant :
   `int quick_sort(int *tab, int size)`. Cette procédure modifie le tableau passé par son pointeur, et fera usage des fonctions et procédures précédentes. Elle sera bien sûr implantée de façon récursive.
7. Vérifier la complexité et comparer les temps d'exécution des deux procédures (en triant un grand tableau par exemple). Retrouvez-vous les résultats attendus en terme de performances ? Commentez.