Initiation à la programmation en C

TP n°9

Antoine Miné

26 avril 2007

Site du cours: `http://www.di.ens.fr/~mine/enseignement/prog2006/`

Tableaux dynamiques de caractères.

Un *tableau dynamique* est une structure de données qui grossit au fur et à mesure des besoins. Le but de l'exercice est de programmer un tableau de caractères (`char`) dynamique. On utilisera naturellement des blocs de mémoire dynamique (`malloc`, `realloc`).

On propose la structure suivante pour représenter un tableau dynamique :

```
struct tab {
  char* mem; /* bloc de mémoire alloué par malloc */
  int   max; /* taille allouée pour mem */
  int   nb;  /* nombre de char effectivement utilisés dans mem, nb<=max */
};
```

À tout moment, le bloc `mem` sera assez grand pour contenir les `nb` caractères du tableau dynamique. Toutefois, on autorise la taille du bloc à être plus grande (`max>nb`). Celui sera utile pour limiter le nombre d'appels à `realloc` (coûteux)...

Exercice 1. Implantation des tableaux dynamiques.
Téléchargez l'en-tête `tab.h` à `http://www.di.ens.fr/~mine/enseignement/prog2006/tab.h`. Programmez dans un fichier `tab.c` les fonctions donc le prototype est donné dans `tab.h`. On rappelle ici le prototype de ces fonctions :

```
struct tab* cree_tab();
void detruit_tab(struct tab* t);
void ajoute_char(struct tab* t, char c);
void ajoute_string(struct tab* t, char* s);
void vide_tab(struct tab* t);
void affiche_tab(struct tab* t);
```

- `cree_tab` alloue une nouvelle structure de type `struct tab` (avec `malloc`), la remplie et la retourne. Initialement, il n'y a aucun caractère (`nb=0`) mais on réservera dans `mem` de la place pour stocker 16 caractères (`max=16`).
- `detruit_tab(t)` détruit `t` (le bloc pointé par `t` et celui pointé par `t->mem`).
- `ajoute_char(t,c)` ajoute le caractère `c` à la fin du tableau dynamique `t`. Si il n'y a pas la place dans `mem` pour le stocker, on augmentera sa taille par `realloc`. Plutôt que de l'augmenter d'un seul octet, on la doublera. Ainsi :
  - pour `nb<=16`, on aura `max=16`,
  - pour `16<nb<=32`, on aura `max=32`,
  - pour `32<nb<=64`, on aura `max=64`, etc.

  et le nombre d'appels à `realloc` est en logarithme du nombre d'appels à `ajoute_char`.
- `ajoute_string(t,s)` ajoute la chaîne `s` (sans le `\0` terminal) à la fin du tableau dynamique.

– `affiche(t)` affiche le contenu du tableau dynamique `t`. Attention, `t->mem` ne se termine pas forcément par un caractère `\0`.
– `vide(t)` vide le tableau dynamique.

Le contrat tacite passé avec l'utilisateur de nos tableaux dynamiques est que :
– tout objet `struct tab*` doit être alloué par `cree_tab`,
– tout objet `struct tab*` doit être détruit par `detruit_tab`,
– après l'allocation et avant la destruction, l'utilisateur peut appeler les fonctions `ajoute_char`, `ajoute_string`, `vide_tab`, `affiche_tab` autant de fois qu'il le veut,
– avant `cree_tab` et après `detruit_tab` il ne doit pas les appeler,
– l'utilisateur peut lire directement les champs d'un objet de type `struct tab*` (contrairement à l'exemple du cours, le type n'est pas abstrait), mais il ne devrait pas les modifier directement (de peur de détruire nos hypothèses sur `mem`, `max` et `nb`).

Exercice 2. Tests.
Écrivez un programme de test `test.c` pour vérifier le bon fonctionnement des fonctions définies dans `tab.c`.

Exercice 3. Systèmes de Lindenmayer.
Programmez une fonction

```
struct tab* subst(struct tab* arg);
```

qui renvoie un nouveau tableau dynamique issu de `arg` où :
– tout caractère `X` de `arg` est remplacé par la chaîne `XgYag`,
– tout caractère `Y` de `arg` est remplacé par la chaîne `daXdY`,
– tout autre caractère de `arg` est recopié inchangé.
Programmez ensuite une fonction

```
struct tab* calcule(int nb);
```

qui part d'un tableau contenant uniquement le caractère `X` et applique `nb` fois de suite la transformation précédente. Attention à détruire les tableaux dynamiques issus de calculs intermédiaires quand ils deviennent inutiles afin d'éviter toute fuite de mémoire.
On a programmé un L-système (ou système de Lindenmayer).

Exercice 4. Courbe du Dragon.
La fonction `calcule` précédente calcule la courbe du Dragon. Afin de la visualiser, il faut interpréter les caractères du tableau dynamique de la manière suivante :
– `g` indique qu'il faut tourner sur soi-même d'un quart de tour à gauche,
– `d` indique qu'il faut tourner sur soi-même d'un quart de tour à droite,
– `a` indique qu'il faut avancer dans la direction courante,
– `X` et `Y` sont ignorés, ils ne servent que dans le calcul.
Écrivez un programme qui affiche la courbe du Dragon sous forme de fichier PostScript. On rappelle (TP2) qu'un fichier PostScrit est un fichier texte d'extension `.ps` commençant par une ligne : `%!postscript`
Les commandes PostScript suivantes seront utiles :
– pour se placer au centre de la page : `297 419 moveto`,
– pour tourner de X degrés : X `rotate`,
– pour avancer dans la direction courante : `1 0 rlineto`,
– pour l'affichage final : `stroke showpage`.