

Graphisme, chaînes de caractères, pointeurs...

Les TPs sont à réaliser seul ou en binôme. La note sera basée non seulement sur la qualité du résultat mais également sur le travail personnel fourni par l'étudiant. Les deux membres d'un même binôme peuvent obtenir une note différente. La note tiendra compte de la lisibilité du code (indentation, commentaires pertinents, choix des identificateurs,...) ainsi que la clarté du code et la facilité avec laquelle il peut être maintenu. Tout code qui ne compile pas ne sera pas considéré.

TP à rendre par courrier électronique (Julien.Narboux@dpt-info.u-strasbg.fr) sous forme de fichier attaché. L'objet du mail devra contenir la mention "TP4 BP". N'oubliez pas de faire figurer votre nom en début de fichier en commentaire.

I) Listes chaînées.

1. Définir une structure afin de représenter une liste simplement chaînée de caractères.
2. Programmer une fonction qui ajoute un élément en début de liste.
3. Programmer une fonction qui supprime le premier élément de la liste.
4. Programmer une fonction qui remplace dans une liste un caractère par une liste de caractères.
5. Programmer des fonctions qui permettent de passer d'une liste de caractères à une chaîne de caractères et vice-versa.
6. Tester vos fonctions.

II) Tortue graphique

Le dispositif graphique virtuel appelé *tortue* a fait la célébrité du langage de programmation Logo (http://en.wikipedia.org/wiki/Logo_programming_language), en particulier pour l'apprentissage de la programmation. Dans ce TP nous allons implanter une tortue en C, puis s'en servir pour tracer des courbes *fractales* (<http://en.wikipedia.org/wiki/Fractal>).

Afin d'effectuer les dessins, nous utiliserons comme au TP précédent la bibliothèque de fonctions graphiques Cairo.

La tortue graphique est un dispositif virtuel qui trace des lignes à l'écran. On l'utilise en lui donnant des ordres de deux types : avancer d'un certain nombre de pas (en traçant le chemin parcouru), et tourner d'un certain angle. Ainsi, pour une tortue initialement dirigée vers la droite, la succession des ordres avancer 100, tourner 90, avancer 50, tourner -45, avancer 30 produit le dessin suivant :



1. Compléter la définition de la structure `tortue` suivante afin qu'elle contienne toutes les informations nécessaires pour une tortue. Notons que le contexte Cairo contient déjà l'information concernant la position de la tortue.

```
typedef struct {
    cairo_t * cr;
    \\ à compléter
} tortue;
```

2. Programmer la fonction `tourner` qui prend en argument un pointeur sur une tortue et un entier n et qui fait tourner la tortue de n degrés.
3. Pourquoi la fonction `tourner` doit-elle prendre en argument un pointeur sur une tortue et pas une tortue tout simplement ?
4. Programmer la fonction `avancer` qui prend en argument un pointeur sur une tortue et une distance et fait avancer la tortue de la distance donnée en tracant le segment correspondant. On pourra utiliser les fonction `cos` et `sin` de la bibliothèque standard ainsi que la fonction `conv` fournie.

```
// Conversion from degrees to radians.
float conv(int a){
    float pi = acos(-1.0);
    return(pi / 180.0 * a);
}
```

5. Spécifier puis programmer une fonction qui dessine un carré d'une largeur donnée à l'aide de votre tortue.
6. Tester vos fonctions en dessinant quelques carrés.

III) La courbe de Von Koch

Notre première utilisation avancée de la tortue consiste à tracer à nouveau la courbe de Von Koch.

1. Spécifier et programmer la fonction `von_koch` qui trace la courbe de Von Koch d'un certain niveau n à l'aide d'une tortue.
2. Tester votre fonction.

IV) L-systèmes

Un système de Lindenmayer, ou L-système, est une description compacte d'une courbe fractale, par la donnée de *règles de remplacement*. Une courbe est décrite par une séquence de caractères, où

- 'F' signifie avancer d'un pas ;
- '+' signifie tourner à gauche (d'un angle fixé) ;
- '-' signifie tourner à droite (du même angle fixé) ;

Ainsi, la courbe de Von Koch de niveau 1 est décrite par $F+F--F+F$, avec un angle de 60 degrés.

1. Programmer la fonction `action_tortue` (a : entier, t : pointeur vers une tortue, car : caractère) qui exécute l'action indiquée par le caractère car étant donné l'angle a (la fonction affiche un avertissement si car est différent de 'F', '+' et '-').
2. Tester votre fonction.
3. Programmer la fonction `sequence_actions_tortue` (a : entier, t : pointeur vers tortue, s : chaîne de caractères) qui exécute dans l'ordre les actions décrites par les caractères de la chaîne s .

4. Tester votre fonction.

Pour engendrer une courbe fractale par un L-système, on produit sa description en caractères par des règles de remplacement d'un caractère par une liste de caractères. Ainsi, la description de la courbe de Von Koch de niveau n est obtenue à partir de F en appliquant n fois la règle de remplacement de F par $F+F--F+F$. On peut représenter les règles par une structure comportant un caractère à remplacer et la chaîne de caractères qui doit le remplacer.

1. Proposer une structure pour représenter les règles.
2. Définir la liste de règles suivantes pour tester :
 - remplacer 'A' par 'B', 'F', '+'
 - remplacer 'B' par '-', 'F'; 'A'
3. En vous inspirant de la première partie proposer une structure qui permet de représenter une liste de règles.
4. Programmer les fonctions qui vous seront nécessaires pour manipuler cette structure de données (lire et comprendre la suite).
5. Spécifier et programmer une fonction `applique_regles_car` qui produit la liste de caractères obtenue par application d'une liste de règles à un caractère. Le caractère est inchangé si aucune règle ne s'applique. Par exemple, avec la liste de règles R ci-dessus : `applique_regles_car(R, 'A')` produit la liste 'B','F','+'.
6. Spécifier et programmer une fonction `applique_regles_liste_cars` qui produit la liste de caractères obtenue par application d'une liste de règles à une liste de caractères. Par exemple, avec la liste de règles R ci-dessus et la liste l représentant 'A','F','B' : `applique_regles_liste_cars(R, l)` produit la liste 'B', 'F', '+', 'F', '-', 'F', 'A'.
7. Spécifier et programmer une fonction `applique_regles_liste_cars_n_fois` qui produit la liste de caractères obtenue par n applications d'une liste de règles à une liste de caractères.
8. Tester vos fonctions.
9. Programmer une fonction qui dessine les segments définis par une liste de règles, un mot de départ, un nombre d'itérations, un angle pour les rotations de la tortue et un angle de départ.
10. Tester votre fonction sur les ensembles de règles suivants :
 - (a) **Règles :** ('F', "F+F-F+F")
Mot initial "F-F-F"
Angle 60
 - (b) **Règles :** • ('S', "F-F-F+F++FF+F-")
 - ('R', "+F-FF-F-F++F+F")
 - ('X', "X-YR-YR+SX++SXSX+YR-")
 - ('Y', "+SX-YRYR-YR-SX++SX+Y")**Mot initial** "SX"
Angle 60
 - (c) **Règles :** • ('X', "XF+Y")
 - ('Y', "XF-Y")**Mot initial** "XF"
Angle 90

V) Tortue avec mémoire

On va maintenant ajouter une fonctionnalité à notre dispositif de tortue graphique : une mémoire. On désire ajouter deux fonctions : une dont le rôle est de mémoriser l'état courant de la tortue, et une autre pour replacer la tortue dans un état précédemment mémorisé. Plus généralement, on veut permettre de mémoriser successivement un nombre arbitraire d'états, et alors la fonction de remplacement mettra la tortue dans le dernier état mémorisé, puis l'avant-dernier, etc.

1. Garder une copie de votre travail, puis changer la définition de la tortue, afin de rajouter la mémoire de la tortue. Pour cela il faudra programmer une structure de données représentant les piles de tortues.
2. Programmer les fonctions `memoriser` et `restaurer` qui respectivement enregistre l'état de la tortue et replace la tortue dans un état précédemment mémorisé. On pourra utiliser les fonctions `cairo_save` et `cairo_restore` qui permettent d'enregistrer et restaurer des contextes Cairo (ces fonctions empilent et dépilent les contextes). Notons qu'en plus du contexte Cairo il faudra aussi sauvegarder l'angle de la tortue.

Pour notre description des actions de la tortue par des caractères, on ajoute :

- '[' signifie mémoriser l'état ;
- ']' signifie replacer la tortue dans l'état précédemment mémorisé.

1. Compléter vos fonctions afin de gérer ces deux nouvelles actions.
2. Tester votre programme avec l'exemple suivant :

Règles : ('F',"FF+[+F[+F]-F+F]-[-F-[F]+F]")

Mot initial :"F"

Angle :30

VI) Pour en savoir plus sur les L-systèmes

Livre : A. Lindenmeyer et P. Prusinkiewicz. The algorithmic beauty of plants, Springer Verlag, 1990.

Sur le web :

http://en.wikipedia.org/wiki/Lindenmayer_system

<http://www.math.okstate.edu/mathdept/dynamics/lecnotes/node12.html>

<http://mathforum.org/advanced/robertd/lsys2d.html>

<http://www.cs.unh.edu/~charpov/Programming/L-systems/>