T.I.P.E : Nœuds

Nous souhaitons aborder la théorie mathématiques des nœuds avec une approche nouvelle, via l'implémentation des nœuds sous forme de structure de donnée en C, dans le but de résoudre des problèmes fondamentaux de cette théorie par une démarche algorithmique: Comment savoir si un nœud peut être démêlé et comment le faire? Comment déterminer si deux nœuds sont les mêmes ?

Voici les articles dont nous nous servons actuellement:

- ENS Rennes: Théorie des noeuds par Lucas Toury
- Nœuds: Genèse d'une théorie mathématiques de Alexei Sossinsky

I. Structure du répertoire

Un fichier texte **`idees.txt`** contient des idées de fonctions à cette heure non implémentées.

Dans le répertoire **`type_noeuds`** se trouve l'implémentation concrète d'une structure de nœud ainsi que les premières primitives pour manipuler la structure de donnée:

`myknot.h`: on y trouvera la définition du type construit **`knot`** et le type des premières primitives. (cf II)

`knot.c`: ce fichier contient l'implémentation de toutes les primitives fondamentales réalisées jusqu'à présent.

`figures.c`: ce fichier contient toutes les fonctions de constructions de nœuds particuliers.

II. Définition du type `knot`

Un *nœud*, dans son sens mathématique, est une ligne fermée dans l'espace. Pour simplifier leur étude, les nœuds sont conventionnellement représentés sur le plan, comme une ligne fermée, en mettant en avant les croisements de cette ligne avec des coupures ce qui distingue la portion du nœud supérieur de la portion inférieure. On appellera cordes par la suite ces portions de nœud. (chaque corde possède une couleur différente dans le 2e exemple)

Exemple de nœuds:

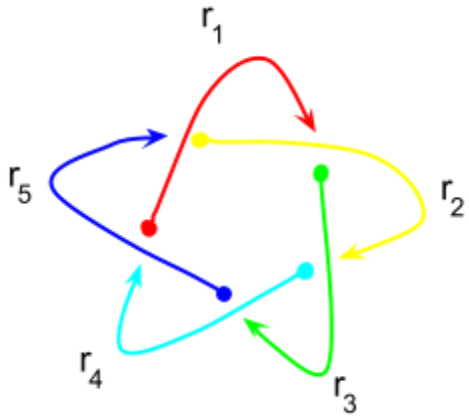
Pour cette implémentation, nous allons donc visualiser le nœud comme la liaison et la superposition des différentes cordes.

Voici la définition récursive du nœud que nous avons trouvé: Un nœud k est une corde r muni de ses attachements à un nœud c_p à une extrémité, et à un nœud c_n à l'autre extrémité. Au dessus de chaque attachement se trouve un nœud de superposition (ou de croisement) x.

On a alors choisi de faire une implémentation des nœuds par maillons chaînés, où chaque maillon r est chaîné à ses quatres maillons c_p, c_n, x_p et x_n. (cf **myknot.h** pour la définition du type **rope** puis **knot**)

On définit alors le **parcours direct** du nœud par la lecture de chaque corde c_n à partir d'une corde de départ r_0. Il prend fin lorsque l'on retombe sur la corde r_0. Ainsi, pour vérifier la propriété de fermeture du nœud, il faut pouvoir retomber sur la corde r_0 au bout d'un nombre fini de lectures. Autrement dit, le parcours direct termine.

Exemple:



.

Ici, les flèches des cordes pointent vers leur corde c_n
On a alors comme parcours direct, en partant de r_1:
r_1, r_3, r_5, r_2, r_4

On retombe bien sur r_1 en regardant la corde c_n de r_4

III. Primitives

1. *Constructeurs*
2. *Accesseurs*
3. *Transformateurs*
4. *Destructeur*

IV. Visualisation ????

V. Tests, Applications.

VI. Approfondissement.